EDIZIONE ITALIANA

APPLE II®
GUIDA ALL'USO

Lon Poole
Martin McNiff
e
Steven Cook



GRUPPO
EDITORIALE
JACKSON

APPLE II®
GUIDA ALL'USO

di
Lon Poole
Martin McNiff
e
Steven Cook

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

Il Gruppo Editoriale Jackson ringrazia per il prezioso lavoro svolto nella stesura dell'edizione italiana la signora Francesca Di Fiore e l'ing. Roberto Pancaldi.

Prima edizione gennaio 1983

Stampato in Italia da:
S.p.A. Alberto Matarelli - Milano - Stabilimento Grafico

SOMMARIO

INTRODUZIONE

Questo libro è una guida all'uso del calcolatore Apple II. In esso troverete sia una descrizione del calcolatore che delle sue periferiche più importanti come unità a disco e stampanti.

Per una lettura più proficua è consigliabile che possiate accedere ad un sistema Apple II già installato e funzionante. Attenzione però che in questo testo non viene descritto come effettuare i collegamenti fisici tra le varie parti di un sistema Apple II in quanto si rimanda ai manuali di installazione, forniti dai costruttori, per la spiegazione di come procedere al montaggio di un sistema Apple II. In quest'opera troverete piuttosto come usare un calcolatore Apple II.

Nel primo capitolo vi diamo una descrizione generale dei vari componenti del sistema Apple II. Mentre nel secondo vi spieghiamo come usarli. A questo punto sarete in grado di potere usare molti dei programmi, già scritti, per qualunque applicazione sia commerciale che finanziaria che per hobby.

Nei successivi quattro capitoli vi insegneremo come programmare in BASIC. Il capitolo 3 è dedicato ai concetti fondamentali della programmazione in BASIC usando sia l'Integer BASIC che l'Applesoft. Il capitolo 4 entra nei dettagli più specifici. Il capitolo 5 è interamente dedicato all'uso delle unità a disco e imparerete così a memorizzare sia programmi che archivi di dati. Il capitolo 6 è dedicato alla grafica sullo schermo che può essere effettuata in due diversi modi.

Il capitolo 7 descrive il programma supervisore Monitor nelle due versioni: Monitor standard e Autostart Monitor. In questo capitolo vedrete anche come inserire un programma scritto in assembler in un altro scritto in BASIC.

Il capitolo 8 è interamente dedicato alla riesposizione di tutte le istruzioni e di tutte le funzioni BASIC in forma di compendio sintetico.

Da ultimo troverete le appendici che di volta in volta sono esplicative di argomenti molto importanti.

CAPITOLO 1

PRESENTAZIONE DELL'APPLE II

Un sistema Apple II si presenta come in Figura 1.1 costituito da più componenti.

Forse il vostro sistema non appare esattamente come quello indicato in quanto ogni sistema Apple II può avere una sua speciale configurazione. In qualunque caso però un sistema Apple II deve comprendere almeno tre parti fondamentali: il calcolatore vero e proprio, la sua tastiera e uno schermo televisivo.

Descriveremo ora le funzioni dei componenti principali senza però dare alcuna indicazione di come installarli. Per queste operazioni vi rimandiamo ai manuali specifici che vengono forniti assieme ad ogni parte del sistema Apple II.

TASTIERA E SCHERMO

La *tastiera* e lo *schermo* televisivo sono gli organi principali che permettono di co-

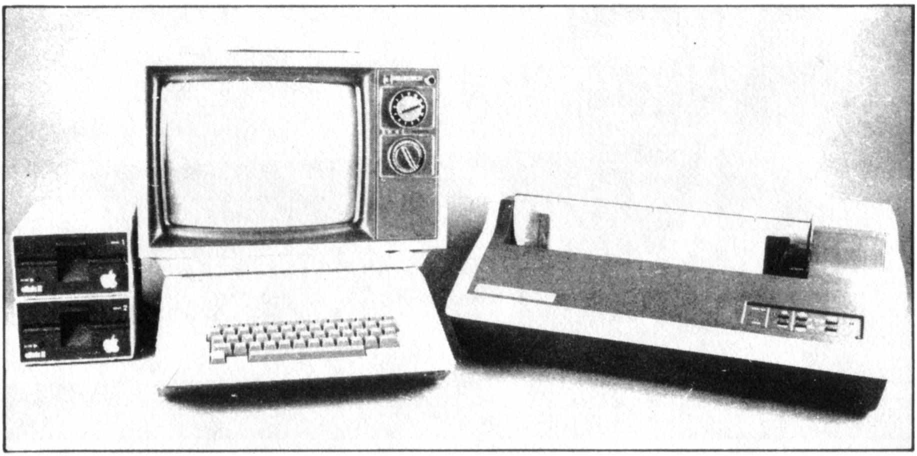


Figura 1.1 — Un tipico esempio di sistema Apple II

municare con il calcolatore. La tastiera assomiglia a quella di una macchina da scrivere e attraverso ad essa l'operatore invia i suoi comandi al calcolatore.

Lo schermo televisivo può essere un normale televisore o un monitor e in ambedue i casi può essere a colori o in bianco e nero. Lo schermo ha la doppia funzione sia di riportare ciò che si batte sulla tastiera, come la carta della macchina da scrivere, sia comunicare all'operatore le risposte del calcolatore.

Uno schermo ha tre diversi modi di operare. Il primo consiste nella rappresentazione di un testo di caratteri in bianco e nero; gli altri due modi riguardano la visualizzazione di grafici.

Nel *modo testo* lo schermo standard viene ripartito in 24 linee ognuna di 40 caratteri (o colonne). *I modi grafici* fanno invece apparire sullo schermo punti o linee, non caratteri, e lo dividono in maniera più o meno fitta (vedere più avanti il capitolo 7).

Come abbiamo già detto è possibile usare un normale televisore (vedi Figura 1.2) come display del nostro sistema Apple II, ma talvolta è preferibile usare quel tipo speciale di televisore che viene chiamato *monitor*. Un monitor è semplicemente un televisore a cui manca la parte elettronica in alta frequenza di ricezione del segnale, ma con il grande vantaggio di produrre una immagine molto più buona. È ovvio che con un monitor non potrete quindi ricevere le normali trasmissioni televisive ... Portobello o Domenica in!

Se usate allora un televisore come display dovete avere due piccoli dispositivi elettronici per il suo collegamento al calcolatore. Il primo dispositivo è posto vicino al-

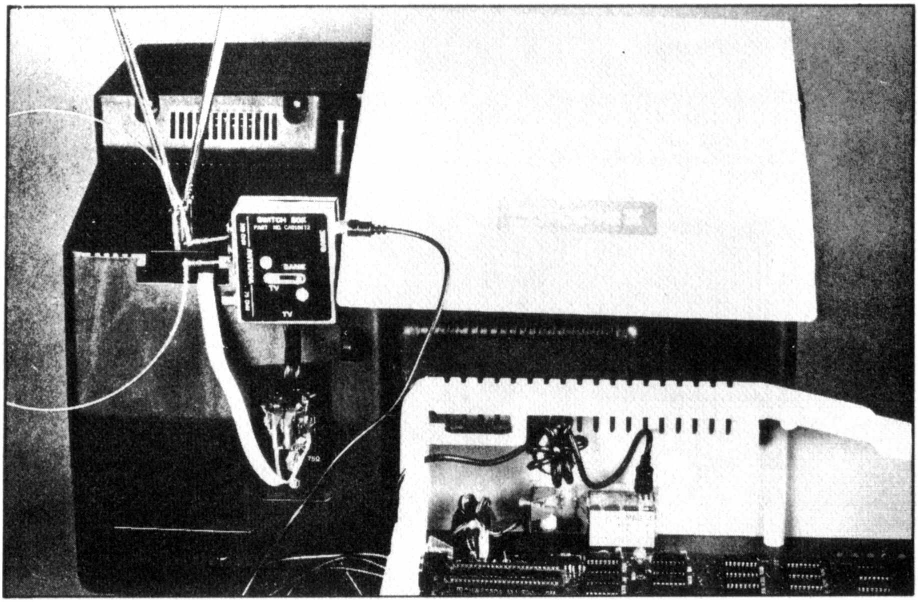


Figura 1.2 — Collegamento con il televisore

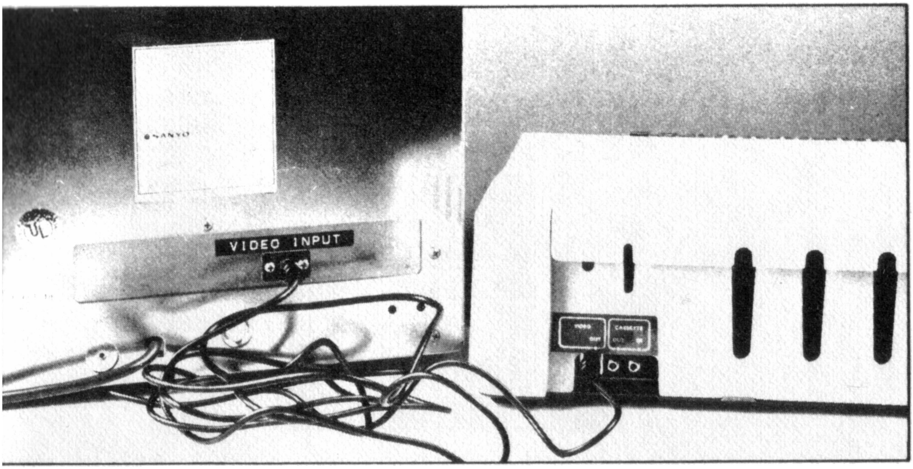


Figura 1.3 — Collegamento con il monitor televisivo

l'antenna e serve per commutare il televisore stesso dall'uso normale con l'antenna all'uso con il calcolatore. Il secondo dispositivo viene posto all'interno del calcolatore ed ha il compito di preparare il segnale, da inviare al televisore, in alta frequenza; tale dispositivo viene chiamato *modulatore in RF.*

Il collegamento con un monitor non richiede questi due dispositivi, ed in particolare il modulatore in RF, perchè esso viene collegato direttamenti al calcolatore (vedi Figura 1.3).

L'INTERNO DELL'APPLE II

Come si può vedere nella Figura 1.4 l'Apple II stesso contiene le parti elettroniche più importanti di tutto il sistema come le memorie, il microprocessore, i punti di connessione delle periferiche, ecc.

Se l'interno del vostro Apple II non è identico a quello riportato nella Figura 1.4 questo è dovuto al fatto che ogni sistema Apple II può essere dotato di cartoline elettroniche diverse montate verticalmente sui connettori (in inglese "slot") posti dietro alla grande cartolina che contiene tutti i circuiti integrati.

MEMORIA

Quasi tutte le memorie dei calcolatori sono misurate in numero di byte. Ogni byte di memoria può memorizzare un carattere. Un calcolatore Apple II può avere da 4096 a 65536 byte di memoria. Comunemente si dice che una memoria è di 4K o 32K o 64K intendendo che un "K" equivale a 1024 byte. Dall'ammontare della memoria totale disponibile dipende poi ciò che il calcolatore è in grado di fare come vedremo meglio in seguito.

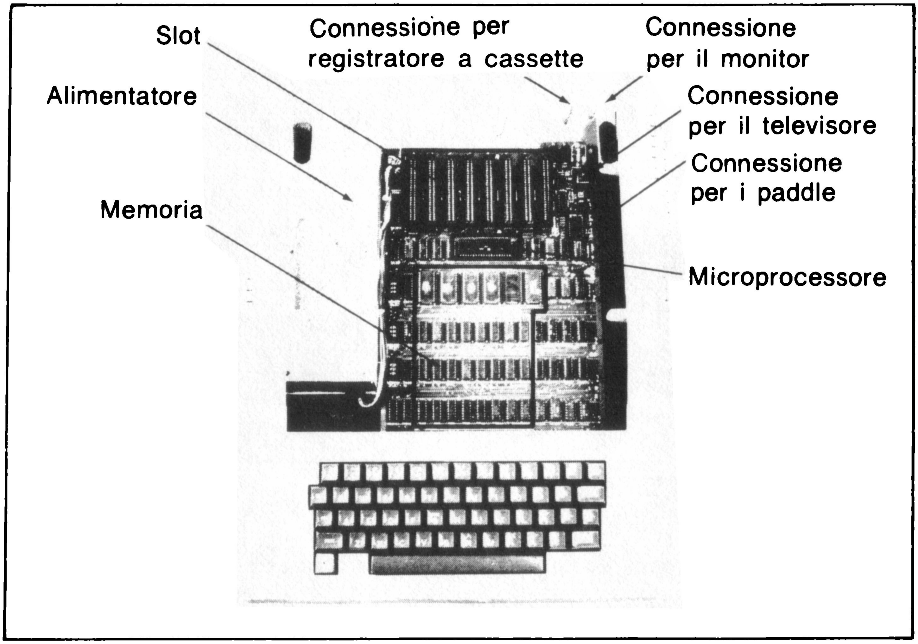


Figura 1.4 — Interno dell'Apple II

È importante dire poi che nel calcolatore Apple II esistono due tipi di memorie. La prima viene chiamata *memoria a sola lettura* ROM (Read Only Memory) ed è quella che contiene tutti i programmi di base per il funzionamento del calcolatore. Essa non può essere modificata ne ri-scritta e il suo contenuto non viene perso spegnendo il calcolatore.

La seconda memoria è quella che conterrà i programmi e i dati dell'utente e viene chiamata *memoria a lettura e scrittura* RAM (Random Access Memory). Questa seconda memoria viene però cancellata totalmente spegnendo il calcolatore.

REGISTRATORE A CASSETTE

L'inconveniente di perdere il contenuto della memoria RAM, spegnendo il calcolatore, è facilmente superato usando una memoria magnetica esterna che nel caso più semplice può essere costituita da un registratore a cassette. In Figura 1.5 viene illustrato un esempio di collegamento tra un calcolatore e un registratore.

UNITÀ A DISCO

Per registrare programmi e dati è possibile anche usare le unità a disco che sono dei registratori magnetici molto più sofisticati ed efficienti di quelli a cassette. Esisto-

no molti tipi di unità a disco con capacità di memorizzazione alquanto diverse tra loro. In Figura 1.6 sono illustrate due unità Disco II (Disk II) della Apple Computer Inc...

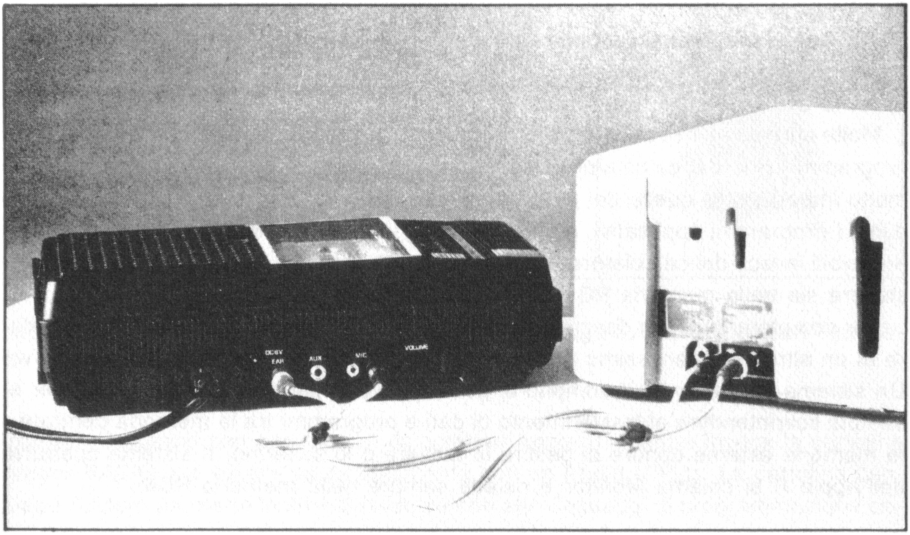


Figura 1.5 — Collegamento con il registratore a cassette

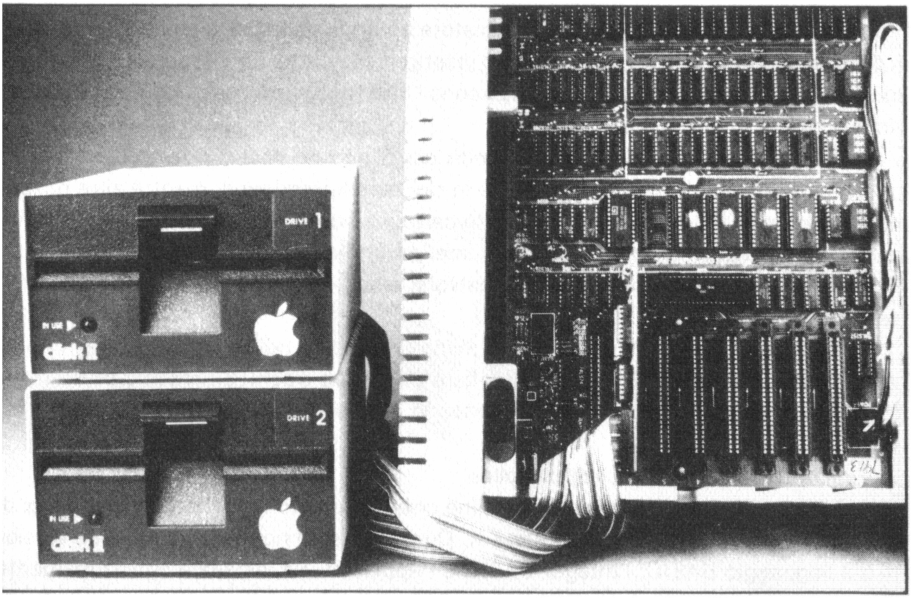


Figura 1.6 — Collegamento con i dischi

PROGRAMMI

È importante che chiariamo subito alcuni concetti relativi alla parola programmazione. Quando si dice che si è scritto un programma, in genere si pensa che si sia scritto un programma per risolvere problemi del tipo commerciale o scientifico o contabile. Questi programmi vengono detti *applicativi.* Essi risiedono nella memoria RAM e possono essere conservati su una cassetta o un dischetto. Nel capitolo 2 vedremo appunto come operare in tal modo.

Molte altre volte invece i programmi non sono di tipo applicativo, ma sono piuttosto programmi usati dal calcolatore stesso per la gestione di altri programmi. Un caso molto importante è quello dei programmi chiamati *interpreti* che permettono di tradurre i programmi applicativi, scritti in linguaggi facili per l'uomo, in linguaggi comprensibili invece dal calcolatore. L'Apple II ha due di questi interpreti che possono risiedere sia nella memoria ROM che in quella RAM.

Per completare questa descrizione generale dei tipi di programmi dobbiamo parlare di un altro importantissimo programma che prende il nome di *sistema operativo.* Un sistema operativo ha il compito di gestire le varie parti del sistema come per esempio sovrintendere al trasferimento di dati e programmi tra la memoria centrale e le memorie esterne oppure di gestire la tastiera o lo schermo. Il sistema operativo dell'Apple II si chiama *Monitor* e risiede sempre nella memoria ROM.

CONTROLLORI PER UNITÀ ESTERNE

Se guardate l'interno del vostro calcolatore noterete nella parte posteriore dei punti di connessione, slot, in cui si possono inserire delle cartoline. Alcune di queste cartoline si chiamano *controllori* e costituiscono l'interfaccia tra il calcolatore e le periferiche esterne come le unità a disco.

Le Apple Computer Inc. produce molte cartoline che possono essere inserite in questi slot. Alcune devono essere poste in slot ben determinanti, mentre altre possono essere poste ovunque. Spesso queste cartoline non sono facilmente identificabili, anche se su di esse è riportato il loro nome, perché esso non è facilmente leggibile. Per capire la funzione di una cartolina talvolta è sufficiente vedere in quale slot è inserita e a quale periferica è collegata.

Come si vede nella Figura 1.6, la cartolina controllore del Disco II risiede normalmente nello slot numero 6 e può comandare uno o due dischi. Se fossero presenti altri due drive si dovrebbe inserire un secondo controllore nello slot numero 5 e così via.

Vediamo adesso altri tipi di cartoline.

In Figura 1.7 sono riportate tre cartoline che riguardano tre aspetti dei linguaggi di programmazione del calcolatore Appe II. Due di esse fanno riferimento a due versioni del linguaggio BASIC: l'Integer BASIC e l'Applesoft. Se installate opportunamente una di queste cartoline nello slot numero 0, il vostro calcolatore potrà passare facilmente da un versione di BASIC all'altra. Le cartoline *Language System* e *Applesoft*

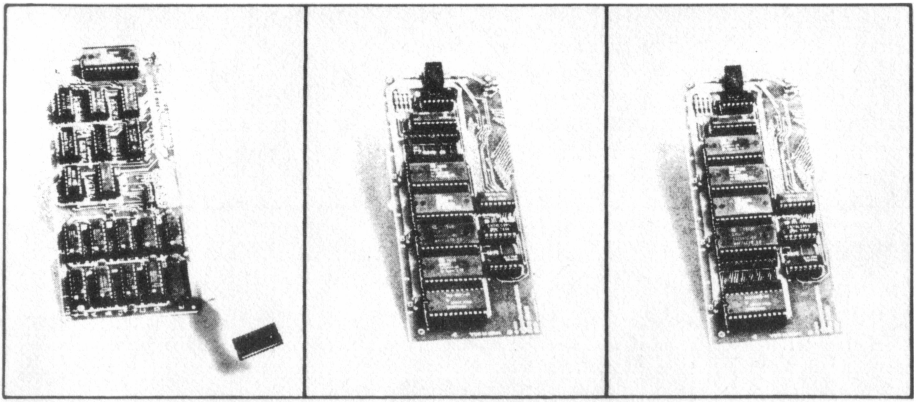


Figura 1.7 — Da sinistra a destra le cartoline: Language System, Applesoft Firmware e Integer BASIC

Firmware lavorano con ogni tipo di calcolatore Apple II, mentre invece la cartolina *Integer BASIC* può essere usata solamente con un Apple II Plus. La cartolina Language System permette inoltre di lavorare con altri linguaggi di programmazione come il Pascal.

La cartolina *interfaccia seriale* della Figura 1.8 risiede normalmente nello slot numero 1; a queste cartolina viene normalmente collegata una stampante.

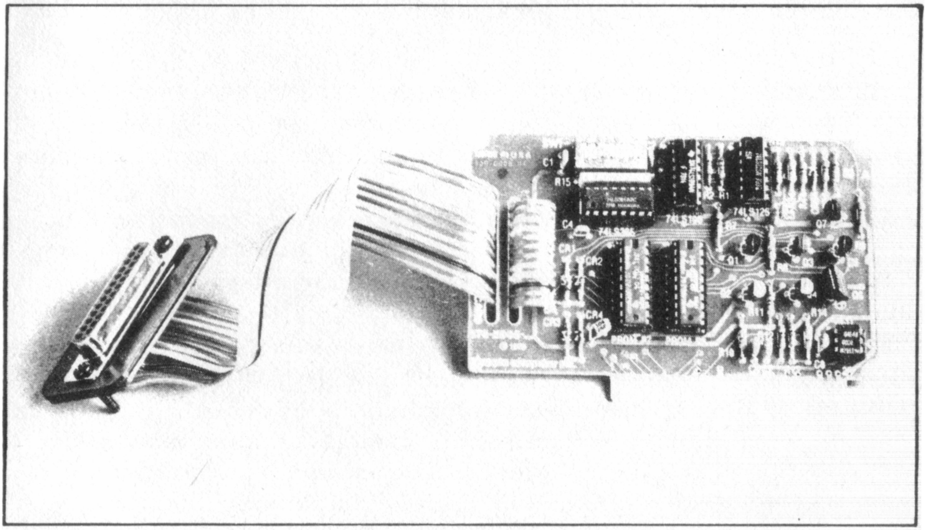


Figura 1.8 — Interfaccia seriale

In Figura 1.9 è visualizzata invece *l'interfaccia parallela per stampanti* che spesso viene usata in alternativa all'interfaccia seriale per collegare le stampanti. Di conseguenza tale cartolina si troverà nello slot numero 1.

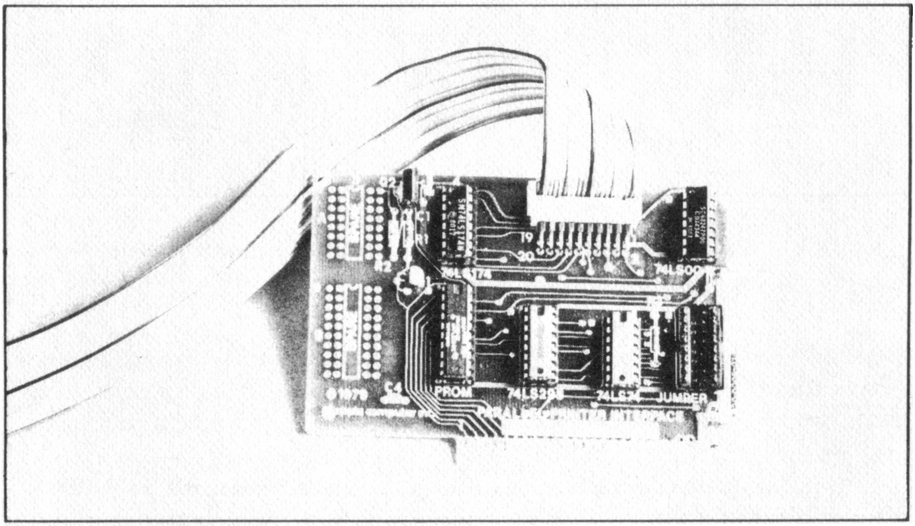


Figura 1.9 — Interfaccia parallela per stampante

La cartolina riportata nella Figura 1.10 è invece *l'interfaccia per comunicazioni remote*. Mediante questa interfaccia e un *modem* potete mettere in comunicazione il vostro Apple II, su una linea telefonica, con un altro calcolatore remoto. (Attenzione però che dovete avere il permesso della SIP!).

Esistono poi moltissime altre cartoline fornite da aziende diverse dalla Apple Computer Inc. Potete trovare cartoline per controllare delle lampadine o della apparecchiature elettriche. È possibile anche usare delle cartoline che permettono di fare la sintesi elettronica di strumenti musicali.

Esiste poi una cartolina estremamente semplice che equivale ad una presa multipla. Essa occupa un solo slot, ma ne mette a disposizione molti altri così da fare una estensione del numero totale di slot disponibili.

Alcuni sistemi Apple II hanno una speciale cartolina che consente di duplicare il numero di caratteri visualizzabili su una riga dello schermo. Questo vale solo però se usate un monitor e non un televisore. In tale caso il monitor deve essere collegato alla cartolina speciale e non alla grande piastra centrale. Spesso questa cartolina deve essere posta nello slot 3 o 4 (in Figura 1.11 è riportato un esempio di tale cartolina).

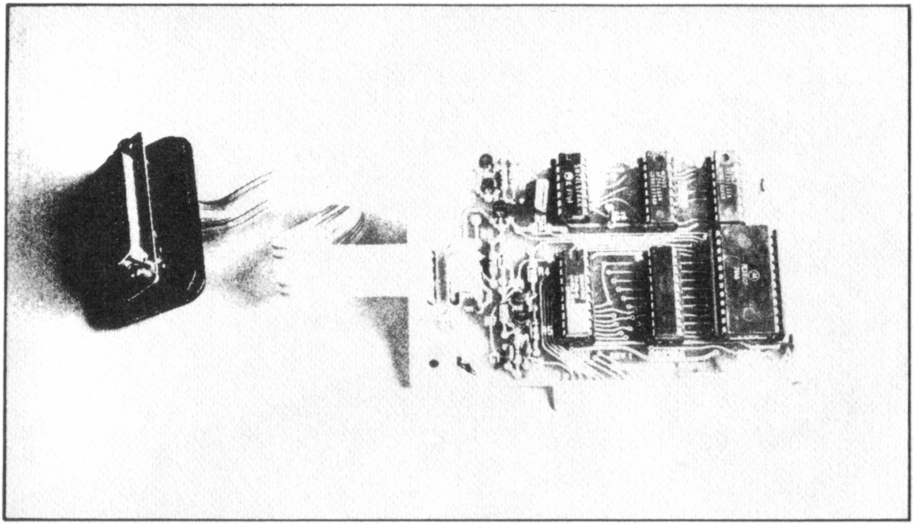


Figura 1.10 — Interfaccia per comunicazioni remote

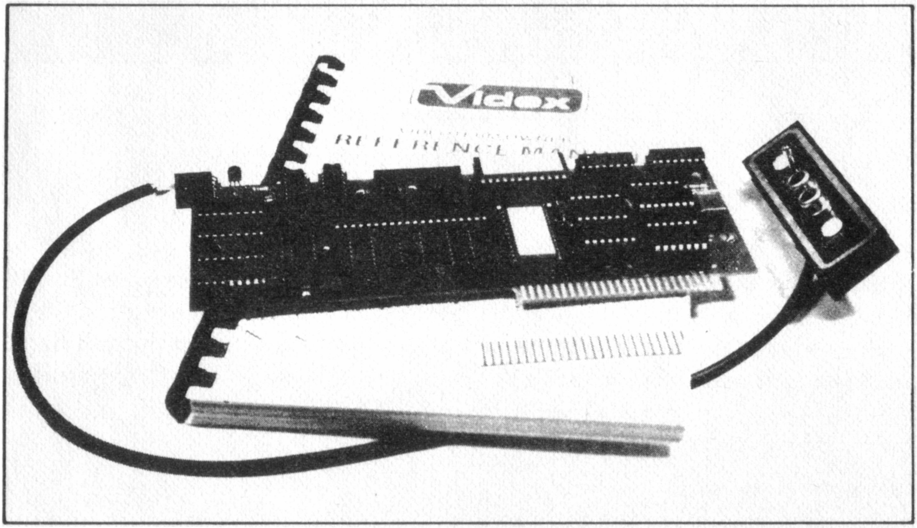


Figura 1.11 — Cartolina speciale per collegamento del monitor

CONTROLLORI PER GIOCHI

Concludiamo questa rapida descrizione dell'interno di un calcolatore Apple II parlando dei controllori per giochi o «paddle» (vedi Figura 1.12). I paddle sono usati essenzialmente dai programmi per giuochi, anche se possono essere impiegati per altre funzioni. Spesso i paddle non sono forniti con il calcolatore.

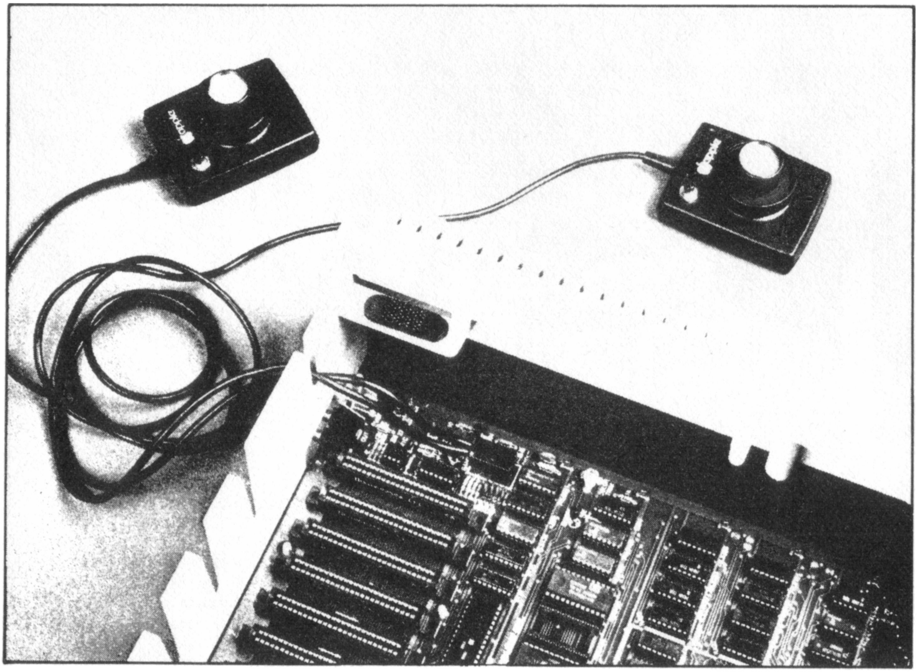


Figura 1.12 — Comandi per i giochi

STAMPANTE

Vediamo ora la stampante come riportato in Figura 1.13.

Essa può essere collegata al calcolatore tramite l'interfaccia seriale o quella parallela a seconda delle sue caratteristiche. Normalmente lo slot usato è il numero 1.

Esistono molti tipi di stampanti alcune semplici come una macchina da scrivere, altre molto costose e complicate. Le stampanti che noi useremo sono di tipo intermedio.

TAVOLETTA GRAFICA

La tavoletta grafica della Apple Computer Inc., illustrata nella Figura 1.14, è un validissimo dispositivo che permette di potenziare notevolmente le caratteristiche grafiche del vostro calcolatore. Mediante la sua *penna* potete disegnare liberamente, sullo schermo anche a colori. I vostri disegni possono occupare tutto lo schermo o solo una parte. Potete poi muoverli, allargarli e separare uno ad uno i loro colori. In ogni momento è possibile poi salvare su disco le vostre immagini per riutilizzarle successivamente. Da ultimo è possibile anche, con la tavoletta grafica, effettuare delle misure di distanze.

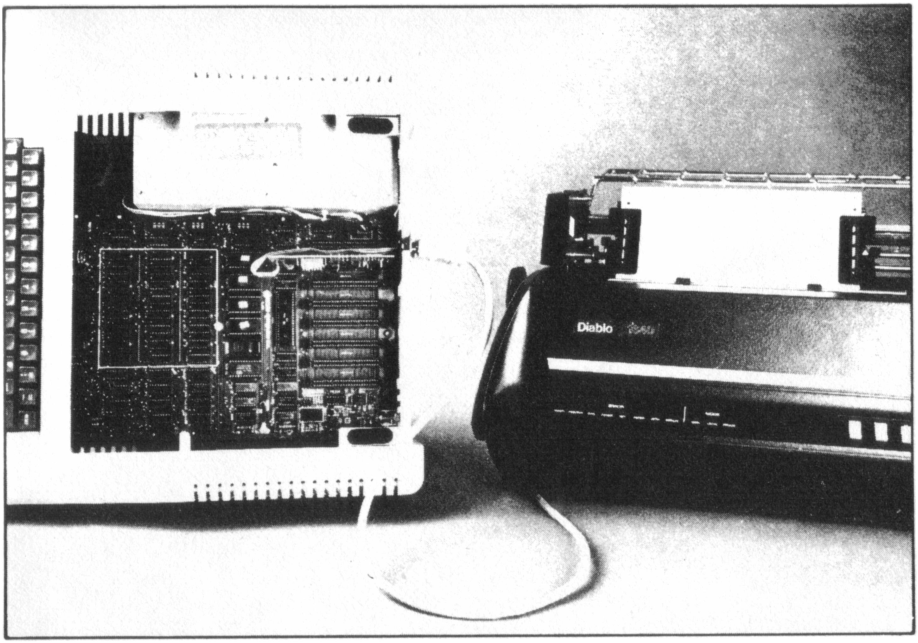


Figura 1.13 — Collegamento della stampante

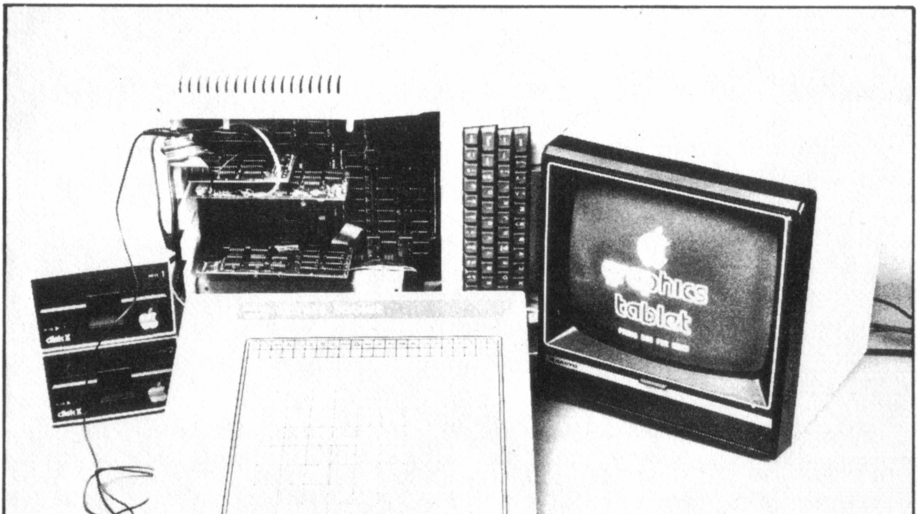


Figura 1.14 — Collegamento della tavoletta grafica

CAPITOLO 2

COME OPERARE CON L'APPLE II

In questo capitolo vi viene descritto come operare con un calcolatore Apple II e cioè come usarlo. Tutto ciò che riguarda invece l'installazione e la manutenzione delle varie parti di un sistema Apple II, non viene qui trattato. Per queste operazioni preliminari vi rimandiamo ai manuali esplicativi che vengono forniti con ogni componente Apple II. Se avete dei dubbi di non aver montato correttamente qualcosa, vi consigliamo di interpellare qualcuno che già conosce bene il calcolatore Apple II.

ACCENSIONE DEL CALCOLATORE

Controllate che tutti i collegamenti siano corretti e quindi accendete il televisore e abbassate al minimo il suo volume. Portate il commutatore, posto sulla scatoletta che si trova appesa all'antenna, nella posizione GIOCHI (o GAME o COMPUTER). Portate il canale del televisore al valore indicato dal modulatore RF (spesso è il canale 33). Se non conoscete il valore di questo canale dovete chiederlo a chi vi ha fornito il modulatore stesso.

Individuate l'interruttore di accensione dell'Apple II, che è posto sul retro vicino al cordone di alimentazione, e accendete il calcolatore. La lampadina POWER sulla tastiera si illumina e il calcolatore vi avvisa di essere pronto a lavorare con un segnale sonoro "beep".

Se non sentite questo "beep" provate a spegnere e riaccendere il calcolatore. Se non lo sentite ancora, e se neanche la lampadina POWER si accende, provate per prima cosa a controllare che la presa di corrente a cui vi siete collegati sia funzionante. Questo lo potete fare facilmente provando ad attaccarvi una lampada o una radio. Se la presa è funzionante, allora spegnete il calcolatore e NON RIACCENDETELO. Forse vi è un collegamento errato o qualcosa è guasto. Prima di accendere ancora il calcolatore chiedete ad un esperto di fare un controllo.

IMMAGINI SUL TELEVISORE

Dopo che avete acceso correttamente il calcolatore e udito il "beep", sullo schermo del televisore appare un'immagine.

Il tipo di immagine che vi appare non può essere qui descritta dettagliatamente perchè dipende dal tipo di Apple II che avete. In ogni caso però, qualunque sia l'immagine che vi appare, sullo schermo potete individuare un piccolo quadratino chiaro che lampeggia con intermittenza: il *cursore.* Il cursore è un segno di riferimento che vi indica dove apparirà il prossimo carattere che batterete alla tastiera. Esso equivale all'indicazione del punto ove battete i caratteri su una normale macchina da scrivere. Nel paragrafo successivo vi indicheremo cosa fare se il cursore manca.

MANCANZA DEL CURSORE

Se il vostro calcolatore ha la cartolina Language System, installata assieme ad una o più unità a disco Disk II, non vedrete apparire il cursore, ma udirete invece un ronzio proveniente dal disco e accendersi la sua lampadina IN USE. In questo caso premete il tasto RESET: il ronzio deve cessare e il cursore apparire.

Esiste un'altra possibilità che il cursore non appaia subito all'accensione del calcolatore e ciò si verifica quando il calcolatore è predisposto per avere più di 40 caratteri per riga. Per far apparire il cursore dovete eseguire questa sequenza di comandi:

1. Mentre tenete premuto il tasto CTRL premete anche il tasto B; rilasciateli quindi contemporaneamente.
2. Per dare il secondo comando dovete prima sapere in quali slot è inserita la cartolina di interfaccia del monitor TV (o 3 o 4). Per saperlo aprite il calcolatore e osservate gli slot che sono numerati da 0 a 7 dalla sinistra verso destra (vedere la Figura 1.11). Se avete dei dubbi chiedete consiglio ad un esperto.
3. Se il vostro monitor TV è collegato allo slot 3 battete PR # 3 e poi il tasto RETURN. Analogamente se è collegato allo slot 4 battete PR # 4, se invece è lo slot 2 battete PR # 2 e così via.
4. A questo punto il cursore deve essere visibile anche se non intermittente. Continuate quindi come indichiamo qui di seguito.

Tabella 2.1 — Caratteri di pronto

Carattere di pronto	Linguaggio
*	Monitor
>	Integer BASIC
]	Applesoft

IL CARATTERE DI PRONTO

Alla sinistra del cursore appare un altro carattere che indica con quale linguaggio il vostro calcolatore lavora in questo momento. Nella Tab. 2.1 sono indicati i tre caratteri di pronto: l'asterisco (*), il simbolo di maggiore di (>) e la parentesi quadra destra (]). Vi ricordiamo che l'Apple II è un calcolatore multilingue e ogni carattere di pronto individua un particolare linguaggio di programmazione.

* è il Monitor

Il carattere di pronto che appare in molte versioni di Apple II è l'asterisco *. Se sul vostro schermo non appare, passate allora alla lettura dei paragrafi seguenti.

L'asterisco vi segnala la presenza del *linguaggio in Assembler Monitor (Assembly Language Monitor)*. Solo se siete però dei programmatori molto esperti vi può interessare di lavorare con il Monitor, diversamente dovete dare il comando per passare dal Monitor al BASIC. Se per divertimento avete dato invece qualche altro comando, sotto il controllo del Monitor, per annullarlo dovete spegnere e riaccendere il calcolatore.

All'apparire quindi dell'asterisco * date il comando per passare al linguaggio BASIC.

CTRL-B per passare in BASIC

Nell'ambito del Monitor esiste un comando per passare al linguaggio BASIC. Questo comando si ottiene battendo contemporaneamente il tasto CTRL e la B (ricordatevi però che il tasto CTRL deve essere premuto un istante prima del tasto successivo). Sotto l'asterisco appare quindi il nuovo carattere di pronto che dipende dal tipo di Apple II che avete e dalle sue opzioni: potrà essere il simbolo di maggiore > o la parentesi quadra].

La parola BASIC è l'acronimo di "Beiginner's All-purpose Symbolic Instruction Code" (Linguaggio di programmazione generale per principianti) e indica un linguaggio di programmazione molto semplice ed efficace sviluppato nell'università americana di Dartmouth. I calcolatori Apple II possono lavorare con due versioni di BASIC, più sviluppate rispetto alla versione originale; l'"Integer BASIC" e l'"Applesoft".

> indica l'Integer BASIC

Il carattere di pronto > indica che il calcolatore può ricevere istruzioni del linguaggio Integer BASIC.

Le due versioni di BASIC, l'Integer BASIC e l'Applesoft, sono molto simili salvo alcune differenze che vi descriveremo in seguito. Per ora vi riportiamo quelle istruzioni che sono valide in ambedue i linguaggi.

] indica l'Applesoft

La parentesi quadra] vi indica che il calcolatore può lavorare con l'Applesoft. Co-

me vi abbiamo già detto, non preoccupatevi per ora di distinguere le due versioni di BASIC. Quando sarà necessario ne riparleremo.

LA TASTIERA

La tastiera del calcolatore Apple II (vedi Figura 2.1) assomiglia molto a quella di una normale macchina da scrivere, salvo avere cinque tasti in più. Due sono sul lato sinistro: ESC e CTRL. Gli altri tre sono sul lato destro e sono marcati con RESET, ← e →. Sul lato destro vi sono poi altri due tasti che forse conoscete già: RETURN e REPT.

Provate liberamente a battere qualche tasto. Qualunque cosa battiate non può danneggiare il calcolatore e può essere annullata spegnendo e riaccendendo il calcolatore.

Noterete subito che tutte le lettere vi appaiono sullo schermo sempre in maiuscolo, sia che abbiate premuto o no il tasto SHIFT. Il calcolatore Apple II sa visualizzare infatti solo le lettere maiuscole, ma se lo ritenete opportuno potete collegare il televisore tramite un dispositivo per generare sia le lettere maiuscole che le minuscole. Esistono anche dei programmi speciali capaci di visualizzare anche le lettere minuscole.

Il tasto RESET

Il tasto RESET è molto importante nei calcolatori Apple II. Quando viene premuto, qualunque cosa il calcolatore stia facendo termina e il controllo passa alla tastiera. Sullo schermo apparirà quindi uno dei caratteri di pronto a seconda del tipo di calcolatore che avete.



Figura 2.1 — Tastiera

Bisogna porre sempre molta attenzione a non premere il tasto RESET per errore, specialmente nel caso in cui un disco sia in funzionamento. Un comando inopportuno di RESET può crearvi dei grossi problemi! Per evitare queste battute accidentali, alcuni calcolatori Apple II prevedono che si debba dare CTRL-RESET invece del solo RESET (vedi più avanti la descrizione del tasto CTRL).

Il tasto RETURN

Qualunque carattere battiate sulla tastiera esso appare sullo schermo e viene anche memorizzato nella memoria. Il calcolatore però tenterà di interpretare le informazioni che gli date, solo quando avrete battuto il tasto RETURN. Questo tasto avvisa infatti il calcolatore che gli avete fornito una linea intera. Subito dopo il calcolatore cancella eventuali caratteri, che fossero rimasti sullo schermo dopo il cursore, e inizia ad interpretare la linea che gli avete appena dato. Se questa linea ha un esatto significato, come istruzione di un programma, allora il calcolatore la esegue. Diversamente vi avvisa, che non può essere interpretata, sia con un "beep" sonoro che con un messaggio sullo schermo:

```
?SYNTAX ERROR
*** SYNTAX ERROR
```

Se vedete comparire un messaggio di errore dovete ribattere la linea possibilmente esatta.

Il tasto SHIFT

In un calcolatore Apple II standard le lettere sono sempre maiuscole per cui è inutile abbassare il tasto SHIFT. In alcuni casi invece il tasto SHIFT abbassato genera caratteri diversi. I caratteri indicati in basso sui tasti della tastiera sono generati con SHIFT alzato, mentre quelli indicati in alto sono ottenuti con SHIFT abbassato.

Adotteremo la convenzione di indicare con SHIFT- i caratteri generati con SHIFT abbassato. Per esempio SHIFT-3 produce il carattere #, SHIFT-N produce ^, SHIFT-P produce @ , ecc. L'unica eccezione è SHIFT-G che non richiama il campanello, ma visualizza ancora G.

Il tasto CTRL

CTRL significa "control". Questo tasto è sempre usato con altri tasti analogamente a SHIFT. CTRL deve essere tenuto abbassato mentre viene battuto l'altro tasto. Anche in questo caso adottiamo la convenzione di indicare con CTRL- quando si deve battere un tasto assieme a CTRL. Per esempio CTRL-B.

Il significato dei comandi CTRL- non viene indicato sulla tastiera salvo una eccezione: CTRL-G. Se provate a battere CTRL-G vi accorgerete infatti che si ottiene il suono "beep" del campanello BELL.

Un esempio di comando CTRL- lo abbiamo già incontrato per richiamare il linguaggio BASIC: CTRL-B. Un altro esempio è CTRL-X che annulla in memoria il testo di

una riga e riporta il cursore tutto a sinistra. I vecchi caratteri rimangono sullo schermo, ma è come se non ci fossero più perchè in memoria sono stati cancellati.

Il tasto ESC

ESC significa "escape" (scappare via). Questo tasto era molto usato agli inizi, con i primi calcolatori, quando le telescriventi erano il terminale più comune.

Il tasto ESC ha molti usi, alcuni dei quali li descriviamo in questo capitolo, altri nel prossimo.

Diversamente dai tasti SHIFT e CTRL, il tasto ESC non deve essere tenuto abbassato mentre viene battuto il secondo tasto, ma bensì deve essere battuto e rilasciato prima di battere il secondo tasto. Questa operazione prende il nome di *sequenza escape.*

Un esempio di sequenza escape è ESC-@. Battete ESC e poi P con SHIFT abbassato (SHIFT-P=@). Vedrete cancellare tutto lo schermo e il cursore portarsi nell'angolo in alto a sinistra. (Nella terminologia dei calcolatori questa posizione iniziale del cursore viene detta "*home*").

I tasti ← e →

I due tasti con freccia sono chiamati *freccetta-sinistra* e *freccetta-destra.*

Questi due tasti, come voi stessi potete constatare, sono molto utili perchè vi permettono di correggere gli errori di battitura e cambiare le informazioni che avete già caricato. Il tasto ← lavora come il tasto per tornare all'indietro di una macchina da scrivere. Ogni volta che lo battete il cursore si sposta di una posizione a sinistra e il carattere sotto il cursore viene cancellato dalla memoria. Provate a battere qualcosa sulla vostra tastiera (per esempio PRINT) e poi battete varie volte ←. La parola PRINT rimane sullo schermo, ma non sarà assolutamente più presente nella memoria dell'Apple II. Se continua a battere ← sino all'inizio della riga vedrete che il cursore salterà a capo e apparirà un nuovo carattere di pronto.

Il tasto → fa invece muovere il cursore verso destra e contemporaneamente porta in memoria il carattere su cui è passato. Questo avviene esattamente come se voi aveste ribattuto uno per uno quei caratteri. Provate infatti a battere una parola, poi spostate il cursore con ← e quindi riportate il cursore a destra con →.

Il tasto REPT

REPT significa ripetere. Se tenendo abbassato REPT premete un altro tasto, questo secondo tasto verrà ripetuto sino a che non rilascerete uno dei due tasti. Esso evita di battere più volte uno stesso tasto quando necessario. Per esempio è molto comodo usarlo assieme a ← per cancellare un lungo testo o assieme a → per ricopiarlo.

Per usare correttamente REPT, premete prima il tasto che desiderate ripetere poi, sempre tenendolo premuto, abbassate REPT. Per terminare la battitura alzate prima REPT e poi l'altro tasto.

Gli altri tasti

Tutti gli altri tasti della tastiera Apple II sono quelli che già conoscete per le lettere, i numeri e i caratteri speciali.

Desideriamo però farvi notare che non dovete confondere il numero 1 con la I e la lettera O con lo zero, come avviene nelle piccole macchine da scrivere. Per essere sicuri di non confondere la O con lo zero, questa cifra viene indicata con uno zero barrato Ø.

IL REGISTRATORE A CASSETTE

Se il vostro sistema Apple II comprende un registratore a cassette potrete allora registrare e leggere programmi su una cassetta magnetica. Spesso potete anzi acquistare cassette che già contengono la registrazione di particolari programmi. Nel capitolo 4 vi indicheremo come registrare e leggere programmi su cassetta.

Uso delle cassette

Fate sempre molta attenzione a maneggiare le cassette. Tenete presente sin da ora che non dovete mai toccare con le mani il nastrino magnetico. Anche se pensate di avere le mani molto pulite, toccandolo rovinereste irreparabilmente il suo strato magnetico.

Appena estratta dal registratore una cassetta deve essere riposta nella sua scatoletta. Ricordatevi poi che le cassette devono essere tenute lontane dal calore, dalla luce intensa, dalla polvere, dall'umidità e dai campi magnetici come quello di un qualunque motore elettrico.

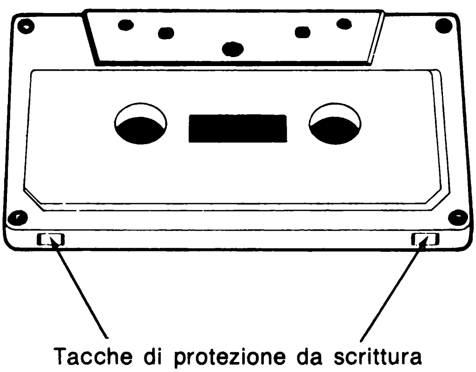


Figura 2.2 — Cassetta magnetica

Etichetta delle cassette

Sicuramente userete molte cassette in cui avrete registrato sia programmi che dati. Per evitare confusione e una inutile perdita di tempo, vi consigliamo di porre un contrassegno, o una etichetta, su ogni cassetta per indicare il suo contenuto.

Protezione delle cassette contro la scrittura

Le cassette magnetiche presentano due fori sul lato opposto a quello del nastro che possono essere ricoperti con due pezzetti di plastica. Questi due fori corrispondono alle due tracce della cassetta. Se sono chiusi potrete sia leggere che scrivere sulla cassetta; se sono aperti potrete solamente leggere. Per proteggere quindi una cassetta, o solamente una sua traccia, da una scrittura accidentale dovete strappare via il pezzetto di plastica dal foro corrispondente. Se in seguito vi interessasse scrivere qualcosa, è sufficiente coprire il foro con del nastro adesivo.

Per sapere a quale traccia corrisponde un foro dovete tenere la cassetta in mano con la faccia che volete proteggere verso l'alto e i fori davanti a voi. Il foro di destra corrisponde alla faccia in alto e il foro di sinistra alla faccia in basso.

Per poter usufruire della protezione contro la scrittura il vostro registratore deve avere il dispositivo meccanico che sente se il foro è aperto o chiuso.

Regolazione del volume del registratore

Per far sì che il vostro registratore funzioni correttamente con l'Apple II dovete regolare il suo volume ad un giusto livello. Se il volume è troppo alto o troppo basso, il segnale utile sarà distorto e il calcolatore Apple II non potrà interpretare le informazioni che gli arrivano dal registratore.

Per determinare quale sia il livello giusto del volume dovete procedere per tentativi. Iniziate con un volume basso e provate a caricare un programma. Se il caricamento non avviene provate con un volume un po' più alto. Continuate così ad alzare a piccoli passi il volume sino a quando constaterete che potete caricare correttamente un programma.

Per prova potete usare una cassetta che ricevete assieme all'Apple II. Se il carattere di pronto che vi appare è], usate allora la cassetta "Color Demosoft". Se il carattere di pronto è >, usate invece la cassetta "Color Graphics".

Inserite la cassetta nel registratore con l'etichetta verso l'alto. Per ogni livello di volume che scegliete, fate quindi così:

1. Riavvolgete completamente il nastro.
2. Battete sulla tastiera dell'Apple II la parola LOAD.
3. Premete il tasto PLAY del registratore.
4. Premete il tasto RETURN.

Appena premuto RETURN il cursore scompare e dopo 15 o 20 secondi il caricamento dovrebbe essere concluso.

Se vi appare il messaggio ?SYNTAX ERROR oppure * * * SYNTAX ERR non toccate il volume, ma tornate al punto 1 e riprovate il caricamento. Se la situazione non cambia allora provate a pulire la testina del registratore o a cambiare cassetta.

Se invece non appare nulla, oppure appare ERR o ERRERR, premete allora RESET, alzate un po' il volume e ricominciate dal punto 1.

Se infine udite un "beep" e non appare nessun messaggio, ciò vuol dire che il caricamento sta avvenendo correttamente. Il calcolatore ha trovato l'inizio del programma e lo sta caricando. Dopo circa altri 15 secondi (a seconda della lunghezza del programma) udirete un altro suono "beep" e sullo schermo appariranno il carattere di pronto e il cursore. Da questo momento il programma è presente nella memoria dell'Apple II.

Prendete nota del livello di volume corretto che avete individuato così da non dover ripetere in futuro questa operazione.

USO DELL'UNITÀ DISCO II

Se invece di avere un registratore a cassette, collegato al vostro Apple II, avete uno o più drive Disco II (Disk II), potete allora leggere i programmi da un dischetto invece che da una cassetta.

Il dischetto "System Master Diskette", fornito assieme ad una unità Disco II, contiene quasi tutti i programmi che potete trovare anche sulle cassette della Apple Computer Inc. più alcuni scritti apposta per il Disco II.

Uso dei dischetti

Quando usate un dischetto dovete fare molta attenzione. I dischetti sono molto più delicati delle cassette. *Non piegate* mai un dischetto, *nè toccate* mai la sua superficie magnetica. *Non forzate* mai l'inserimento di un dischetto in un drive. Appena estraete un dischetto da un drive, inseritelo subito nella sua busta. *Tenetelo lontano* dal calore, dalla luce intensa, dall'umidità e dai campi magnetici (ricordate che vicino ad un motore elettrico c'è sempre un forte campo magnetico). Prestate molta attenzione specialmente al "System Master Diskette".

Come inserire un dischetto in un drive Disco II

Nella Figura 2.3 vi indichiamo come inserire correttamente un dischetto in un drive Disco II. Tenete il dischetto tra il pollice e l'indice, aprite il portellino del Disco II e inserite con delicatezza il dischetto nella fessura orizzontale. L'inserimento non deve presentare alcuna resistenza! Se il dischetto non entra facilmente estraetelo subito e riprovate. Cercate di tenerlo sempre il più possibile orizzontale. Quando il dischetto è inserito nel drive chiudete con delicatezza il portellino. Se incontrate una qualche re-

sistenza riapritelo e provate a spingere bene in fondo il dischetto; quindi riprovate a chiudere il portellino. Se forzate sia l'inserimento del dischetto che la chiusura del portellino potete rovinare seriamente il Disco II. Un consiglio pratico è quello di chiudere il portellino qualche attimo dopo che il dischetto è stato posto in rotazione così che si sia ben centrato sul drive da solo.

IL SISTEMA OPERATIVO A DISCO

Prima di poter usare i dischetti dovete aver caricato in memoria uno speciale programma chiamato "Disk Operating System" (DOS) che sovrintende a tutte le operazioni con le unità a disco. Il processo di caricamento di una copia del DOS in memoria viene detto "booting". Nella terminologia dei calcolatori si dice *fare il booting del DOS*.

Il sistema DOS deve essere caricato ogni volta che il calcolatore viene acceso.

Caricamento del DOS

Il caricamento del DOS può essere fatto in modi diversi a seconda della configurazione del vostro calcolatore e del linguaggio che usate. In ogni caso però è sempre previsto che il drive usato sia collegato al controllore posto nello slot numero 6 e al

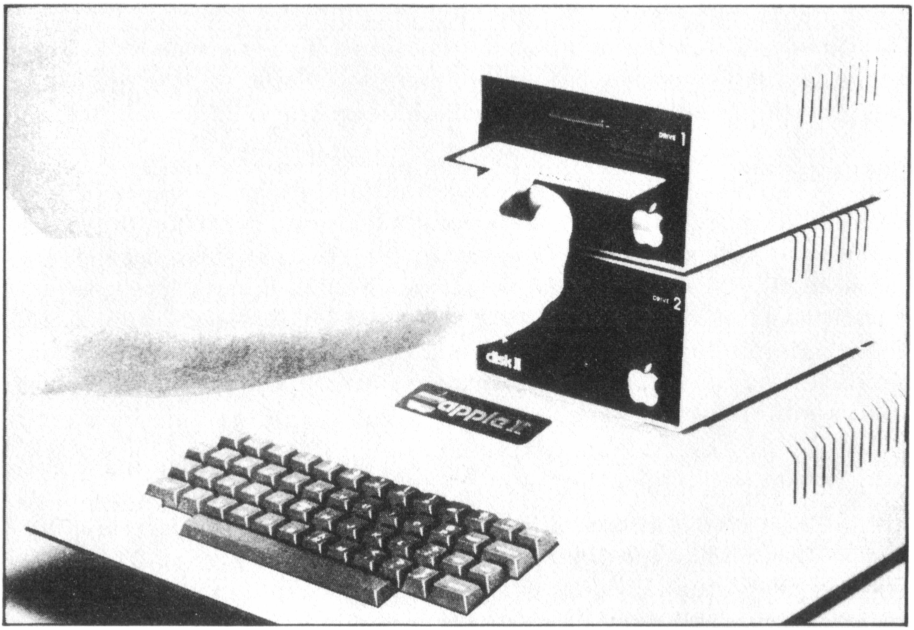


Figura 2.3 — Inserimento di un dischetto nel Disco II

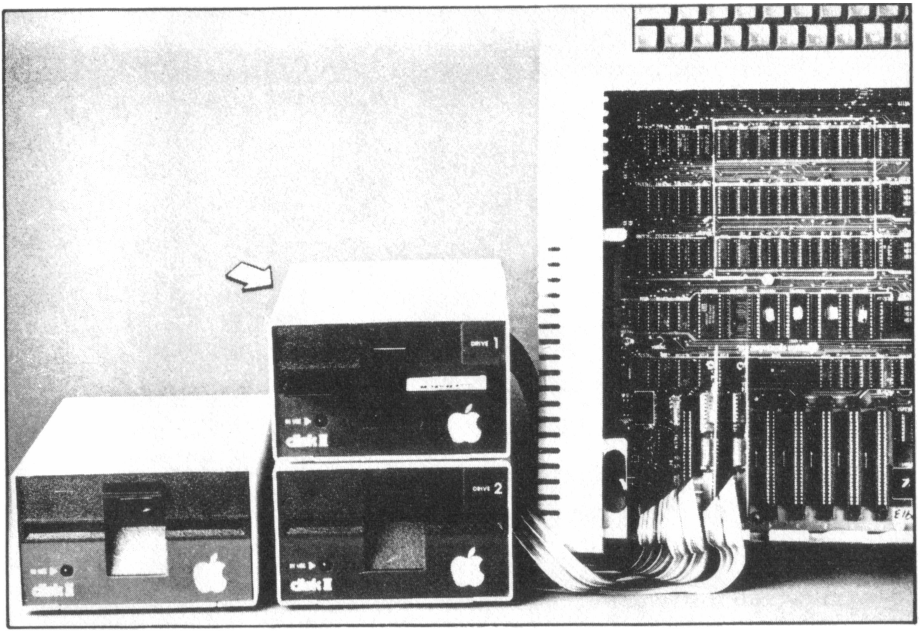


Figura 2.4 — Drive standard per il "booting" del DOS

connettore numero 1 del controllore stesso. Osservate per esempio il drive indicato con una freccia nella Figura 2.4.

Inserite quindi il "System Master Diskette" nel drive che vi abbiamo indicato e chiudete il suo portellino. Alcune volte, se nel vostro calcolatore è inserita la cartolina "Language System", dovete usare il dischetto "Integer and Applesoft II", prima di quello del DOS. Questo caso particolare verrà descritto più avanti nel paragrafo che tratta il caricamento con il "Language System".

Al termine del caricamento del Dos, con uno dei metodi che descriveremo qui di seguito, lo schermo apparirà come indicato in Figura 2.5.

Booting automatico

Se il vostro calcolatore contiene l'"Autostart Monitor", il caricamento del DOS può avvenire automaticamente. In questo caso, appena accendete l'Apple II, vedrete il Disco II mettersi in funzionamento e la lampadina rossa IN USE accendersi.

Nel caso quindi che avete l'Autostart Monitor e che manchi la cartolina Language System, la procedura per caricare il DOS è la seguente. Tenete spento il calcolatore e inserite il dischetto System Master nel drive opportuno. Accendete quindi il calcolatore. Dopo pochi secondi il disco si fermerà e sullo schermo apparirà uno dei messaggi riportati nella Figura 2.5.

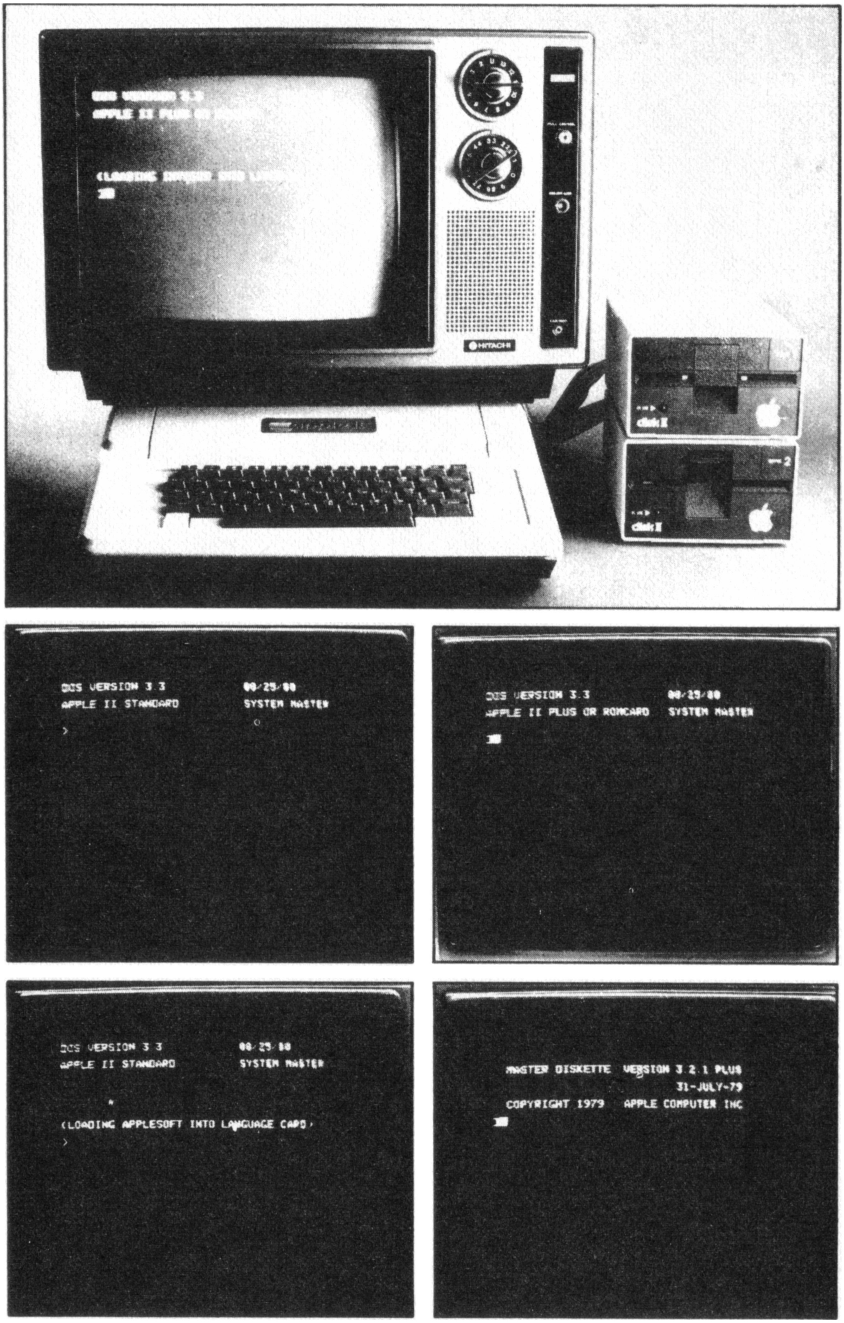


Figura 2.5 — Caricamento corretto del System Master Diskette

Booting sotto il controllo dell'Assembly Language Monitor

Potete caricare il DOS sotto il controllo del linguaggio Monitor quando sullo schermo vi sia apparso il carattere di pronto *. Le sequenze di comandi che potete dare sono due e vengono descritte qui di seguito.

Booting con una istruzione di salto

Se il drive 1 è collegato allo slot 6 (vedi Figura 2.4), potete caricare il DOS con il seguente comando del Monitor:

```
*C600G
```

Ricordatevi di premere RETURN al termine dell'istruzione. La lampadina IN USE si deve accendere e dopo qualche secondo lo schermo apparirà come uno dei casi della Figura 2.5.

Booting con un comando CTRL-P

L'altra sequenza che potete seguire prevede che battiate il numero dello slot a cui è collegato il drive (generalmente 6) e quindi il comando CTRL-P seguito da RETURN. Dopo qualche secondo il DOS sarà in memoria e lo schermo vi apparirà come in Figura 2.5.

Booting dall'Integer BASIC o dall'Applesoft

I comandi di booting sono gli stessi sia per l'Integer BASIC che per l'Applesoft.

Dopo il carattere di pronto (> per l'Integer BASIC e] per l'Applesoft) battete IN oppure PR, il carattere # e il numero di slot. Ecco due esempi:

```
IN#6
PR#6
```

Premete quindi RETURN e attendete alcuni secondi per vedere apparire sullo schermo uno dei casi della Figura 2.5.

Booting sotto il controllo del "Language System"

Quando siete sotto il controllo del "Language System" il caricamento del DOS può avvenire con un passo solo oppure con due a seconda della versione di DOS che avete.

La versione di DOS è un numero, che trovate indicato sul "System Master Diskette", analogo a 3.3 oppure 3.2.1 oppure 3.2.

Nel caso del DOS versione 3.3 il suo caricamento può essere effettuato con uno dei metodi più sopra indicati, indipendentemente che sia presente oppure no la cartolina "Language System". Usate quindi il "System Master Diskette" con una delle pro-

cedure viste più sopra. Automaticamente saranno disponibili anche l'Integer BASIC e l'Applesoft se è presente anche la cartolina "Language System". Quando vedrete fermarsi il disco, il caricamento sarà terminato e lo schermo sarà come in Figura 2.5.

Il caricamento del DOS nelle versioni 3.2 e 3.2.1. richiede invece due dischetti. Il primo dischetto, che dovete caricare con uno dei metodi visti sopra, è denominato "Integer and Applesoft II". La lampadina IN USE si accenderà e il drive si porrà in funzionamento. Dopo alcuni secondi sullo schermo apparirà il messaggio:

```
INSERT BASIC DISK AND PRESS RETURN
```

Aprite allora il portellino del drive, togliete il dischetto, inserite il secondo dischetto "System Master Diskette" e chiudete il portellino. Premete allora RETURN. Dopo alcuni secondi il DOS sarà completamente caricato e sullo schermo vi apparirà uno dei casi della Figura 2.5.

Come leggere il catalogo di un dischetto

Al termine del caricamento del "System Master Diskette" vi interesserà sapere quali programmi questo dischetto contiene. Questa informazione vi può essere data battendo sulla tastiera il comando:

```
CATALOG
```

e quindi RETURN. Sullo schermo vi apparirà qualcosa che assomiglia ad un indice:

```
DISK VOLUME

*I 002 HELLO
*I 053 APPLE-TREK
*I 018 ANIMALS
*B 009 UPDATE 3.2.1
*I 014 COPY
*I 009 COLOR DEMO
*I 053 BRICK OUT
*I 026 SPACE WAR
*I 050 THE INFINITE NO. OF MONKEYS
*I 051 COLOR SKETCH
*I 053 SUPERMATH
*I 026 APPLEVISION
*I 017 BIORHYTHM
*I 027 PINBALL
```

```
NEW

10   REM INIZIALIZZATO IL data

20   REM SISTEMA CON MEMORIA DI byte

30   REM VERSIONE DOS numero

40   PRINT "nome disco"

50   END
```

Figura 2.6 – Esempio di programma iniziale

Caricamento di altri dischetti

La procedura per caricare un qualunque dischetto, preparato per una unità Disco II, è esattamente la stessa vista più sopra per caricare il "System Master Diskette".

Vi ricordiamo però che se un dischetto è stato preparato per un altro calcolatore o per un altro tipo di drive, non può essere caricato nel vostro Apple II.

PREPARAZIONE DI DISCHETTI NUOVI

Prima di poter usare un dischetto nuovo con il vostro sistema Apple II, dovete "inizializzarlo".

A questo punto è possibile però che il programma che state facendo girare con il nuovo dischetto contenga già le istruzioni di inizializzazione. In tal caso non dovete far altro che porre in esecuzione il programma. Se invece esso non contiene le istruzioni di inizializzazione dovete seguire la procedura che vi descriviamo.

Il processo di inizializzazione prepara un dischetto per il suo uso successivo. Durante questa fase qualunque programma sia presente nella memoria dell'Apple II viene trasferito sul dischetto e diviene il programma *di saluto (o iniziale)*. Questo particolare programma verrà sempre caricato ogni volta che fate il booting del dischetto. Un esempio di programma di saluto viene indicato nella Figura 2.6, ma voi potete usare un qualunque altro programma. Si può comprendere allora come sia consigliabile inserire nel programma di saluto tutti quei dati che caratterizzano il dischetto. Nel programma della Figura 2.6 è previsto infatti che inseriate la *data*, il numero di *byte* della vostra memoria (32K, 36K, ecc.), il *numero* della versione del DOS e il *nome* da attribuire al dischetto. Ecco un esempio:

```
NEW
10 REM INIZIALIZZATO IL 2/1/81
20 REM SISTEMA CON MEMORIA DI 48K
30 REM VERSIONE DOS 3.2.1
40 PRINT "MISC., VOL. 3"
50 END
```

Per inizializzare un dischetto dovete quindi preparare un programma di saluto, inserire nel drive il dischetto e battere il comando:

```
INIT HELLO
```

Premete allora RETURN. Il drive entra in funzionamento e per circa due minuti lo sentirete girare. L'inizializzazione terminerà quando la lampadina IN USE sarà spenta.

Ancora un consiglio: preparate una etichetta che descriva il contenuto del nuovo dischetto e applicatela sopra al dischetto stesso.

CARICAMENTO ED ESECUZIONE DI UN PROGRAMMA

Per lavorare con il calcolatore Apple II esistono già molti programmi; alcuni su dischetto, altri su cassetta altri ancora sono registrati sia su cassetta che su dischetto.

Attenzione però che questi programmi possono essere stati scritti o in Integer BASIC o in Applesoft, e che generalmente devono girare con lo stesso tipo di linguaggio.

COME USARE LA GIUSTA VERSIONE DI BASIC

Sebbene la maggior parte dei calcolatori Apple II abbia ambedue le versioni di BASIC, alcuni calcolatori ne hanno una sola. L'Apple II Plus non ha l'Integer BASIC salvo che sia provvisto della cartolina Language System o di quella Integer BASIC. Analogamente l'Applesoft non è presente in molti calcolatori a meno che voi non lo carichiate da cassetta o da disco.

Se il vostro sistema Apple II ha la cartolina Language System oppure quella Applesoft Firmware, la scelta della versione di BASIC avverrà automaticamente per ogni programma che caricate. In ogni altro caso dovrete porre molta attenzione che il carattere di pronto sia quello corrispondente al tipo di BASIC del programma che caricate.

Con il calcolatore Apple II standard è molto facile ottenere il carattere di pronto dell'Integer BASIC (>); basta premere il tasto RESET, poi CTRL-B e quindi RETURN.

Se volete ottenere invece il carattere di pronto dell'Applesoft (]) con un calcolatore Apple II standard, dovete prima aver caricato l'interprete Applesoft da cassetta o da dischetto.

In tal caso dovete dapprima caricare il DOS come vi abbiamo già indicato in questo capitolo. Quando il DOS sarà presente nella memoria del calcolatore, potrete allora caricare l'interprete Applesoft inserendo il System Master Diskette nel Disco II e battendo sulla tastiera il comando:

```
FP
```

Per qualche secondo il drive sarà in funzionamento, poi sullo schermo apparirà il carattere di pronto dell'Applesoft (]).

Se l'interprete Applesoft è depositato su una cassetta invece che su un dischetto, dovete inserire la cassetta "Applesoft II" (della Apple Computer Inc.) nel registratore, quindi riavvolgerla completamente e battere il comando:

```
LOAD
```

Prima di battere RETURN premete il tasto PLAY del registratore. Udirete un primo "beep" poi dopo uno o due minuti un secondo "beep". Fermate quindi il registratore; sullo schermo vi apparirà il carattere di pronto dell'Applesoft.

È possibile passare dall'Applesoft all'Integer BASIC con il seguente comando:

```
INT
```

Attenzione però: se volete tornare all'Applesoft dovete ricaricarlo da dischetto o cassetta.

COME CARICARE UN PROGRAMMA DA UNA CASSETTA

Selezionate la versione di BASIC con cui è stato scritto il vostro programma (questa selezione può essere automatica come abbiamo indicato più sopra), eseguite quindi la seguente procedura:

1. Riavvolgete il nastro sino all'inizio del programma. Se il programma è il primo della cassetta dovrete ovviamente riavvolgerla tutta; se invece il programma che vi interessa è preceduto da altri programmi dovete caricarli tutti uno per uno ripetendo questa stessa procedura più volte.

2. Battete sulla tastiera:

```
LOAD
```

3. Premete PLAY.

4. Premete RETURN.

Appena premuto RETURN il cursore scompare e dopo alcuni secondi udirete un "beep" per indicare che il calcolatore inizia a caricare il programma. Dopo un certo tempo udirete un secondo "beep" per segnalare che il programma è stato completamente caricato; premete allora STOP per fermare il nastro. Se non udite i "beep" oppure se vi viene dato un segnale di errore, provate a controllare il livello del volume come vi abbiamo più sopra illustrato. Se le difficoltà di caricamento permangono, allora è probabile che la cassetta sia rovinata per cui dovete cambiarla.

COME CARICARE UN PROGRAMMA DA UN DISCHETTO

Quando siete sotto il controllo del DOS, cioè lo avete già caricato in memoria, potete caricare un programma da disco mediante il seguente comando:

LOAD *nome programma*

nome programma è ovviamente il nome del programma che volete caricare e deve essere presente sul dischetto che state leggendo.

ESECUZIONE DI UN PROGRAMMA

Per porre in esecuzione un programma caricato in memoria dovete battere il comando:

RUN

In tal caso il controllo del calcolatore, compreso quello della tastiera e dello schermo, passa alle dipendenze del programma. Per riprendere il controllo è spesso sufficiente che battiate CTRL-C eventualmente seguito da RETURN. Se non lo ottenete ciò può dipendere da qualche particolare istruzione del vostro programma. Nel caso peggiore premete RESET oppure spegnete e riaccendete il calcolatore, ma attenzione che in ambedue questi casi potete perdere il vostro programma.

REGOLAZIONE DEI COLORI DEL TELEVISORE

Il sistema Apple II vi permette di tracciare grafici a colori. Per questo motivo dovete regolare i comandi del colore del vostro televisore (o del vostro monitor) per una corretta visualizzazione. Per effettuare tale regolazione potrete usare uno dei programmi già predisposti. La cassetta "Color Graphics" con l'Integer BASIC; la cassetta "Color Demosoft" con l'Applesoft oppure il programma COLOR DEMOSOFT da dischetto.

Caricate quindi e ponete in esecuzione uno di questi programmi. Sullo schermo apparirà il testo indicato in Figura 2.7.

Questo tipo di testo viene più esattamente definito *menù* perchè riporta alcune possibilità di scelta. Nel nostro caso scegliamo la voce 1 per cui battiamo 1 e RETURN. Sullo schermo appariranno 16 strisce verticali colorate (vedi Figura 2.8). Tali

strisce devono avere i seguenti colori per cui dovete regolare i comandi del televisore così che essi siano il più vicino possibile a quelli indicati:

BLAK	nero	BROWN	bruno
MGTA	magenta	ORNG	arancione
DBLU	blu scuro	GREY	grigio
PURP	porpora	PINK	rosa
DGRN	verde scuro	LGRN	verde chiaro
GREY	grigio	YELO	giallo
MBLU	blu medio	AQUA	acqua
LBLU	blu chiaro	WITE	bianco

Cercate di regolare, con attenzione particolare, i colori porpora, rosa, giallo e le tre tonalità di blu.

Al termine della regolazione premete RETURN. Il menù riappare e potete quindi scegliere un'altra funzione. Per uscire da questo programma dovete premere CTRL-C (e RETURN).

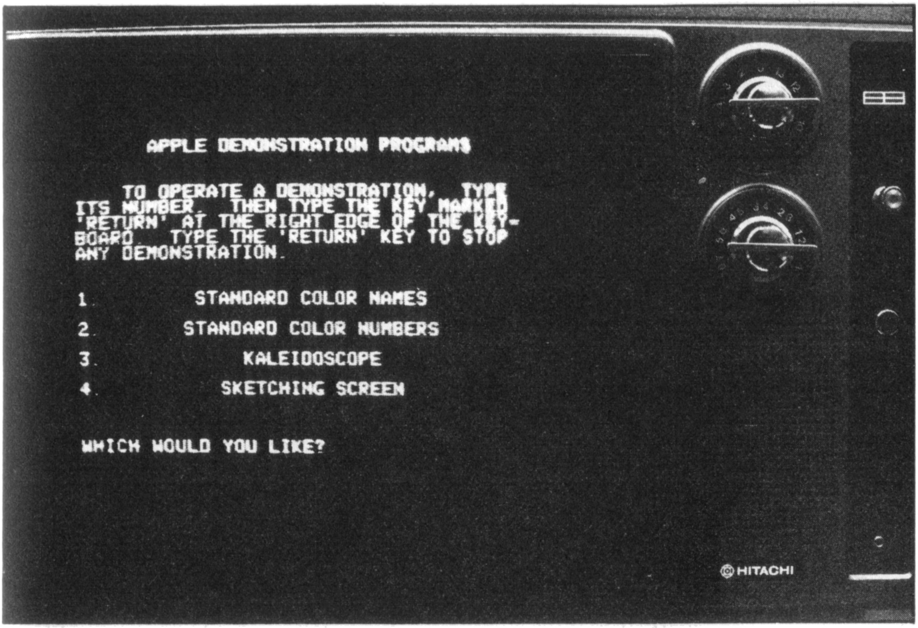


Figura 2.7 — Menù del programma COLOR DEMOSOFT

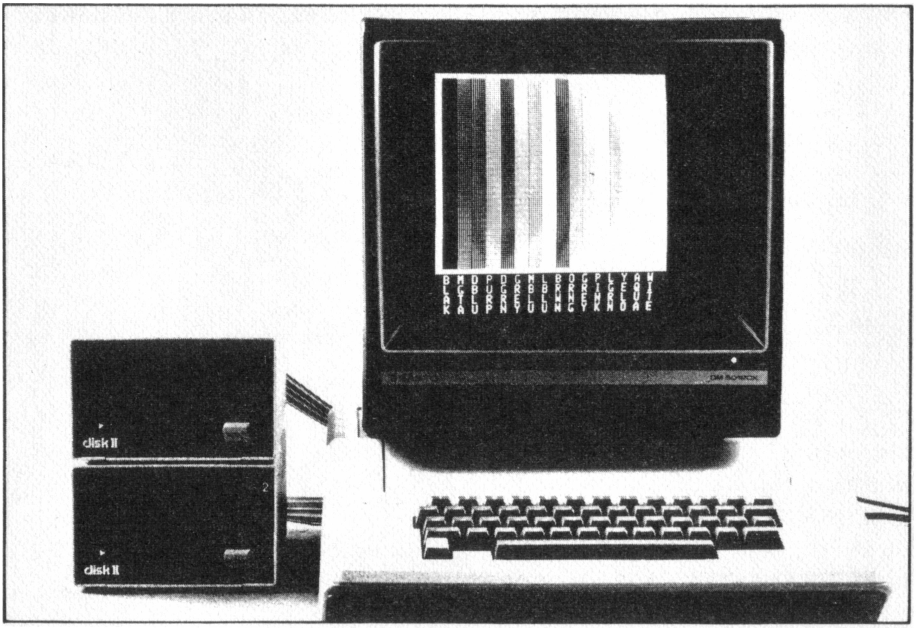


Figura 2.8 — Regolazione del colore

ALTRI COMPONENTI DEL SISTEMA APPLE II

Se il vostro sistema Apple II comprende altri componenti, come per esempio una stampante, non possiamo darvi le istruzioni di funzionamento perchè ogni marca di periferica può differire moltissimo da un'altra. In questi casi dovete sempre leggere attentamente il manuale che vi viene fornito assieme alla periferica.

GESTIONE DEGLI ERRORI

Il sistema Apple II, come tutti i calcolatori, è in grado di eseguire istruzioni e programmi che siano però assolutamente privi di errori. Le conseguenze di un errore, anche minimo, possono falsare tutti i risultati di un intero programma.

MESSAGGI DI ERRORE

Ogni volta che battete una istruzione e alla fine premete RETURN, il calcolatore esamina quanto gli avete detto e spesso può accorgersi se avete commesso un errore. In tal caso suona un "beep" e sullo schermo appare un *messaggio di errore (error*

message). Molte volte questo messaggio vi spiega chiaramente l'errore commesso, ma altre volte ciò non è possibile. In qualunque caso l'unico modo per correggere l'errore è quello di ribattere l'istruzione. Nell'Appendice C trovate una lista completa dei messaggi di errore.

CORREZIONE DEGLI ERRORI DI BATTITURA

Ogni volta che battete una istruzione, o un testo alla tastiera, è molto facile che commettiate degli errori di ortografia. Se ve ne accorgete prima di battere RETURN avete allora molte possibilità per correggerli con alcuni tasti che abbiamo già descritti all'inizio:

- Il tasto ← di ritorno del cursore permette di cancellare i caratteri su cui passa sopra (anche se essi rimangono visibili sullo schermo).
- Il tasto → muove il cursore in avanti e rimemorizza i caratteri su cui passa sopra.
- Il tasto REPT vi permette di muovere le due freccette con maggior velocità.
- CRRL-X cancella completamente la linea su cui state lavorando.
- ESC-@ pulisce tutto lo schermo e riporta il cursore in alto a sinistra.

Vediamo qualche esempio. Supponiamo di dover battere l'istruzione di caricamento:

```
LOAD COLOR DEMOSOFT
```

e invece abbiamo battuto:

```
LOAS COLOR DEMOSOFT
```

Abbiamo due possibilità per correggere l'errore: con CTRL-X cancellare tutta la riga e poi ribatterla, oppure spostare a sinistra il cursore con ←. Proviamo con il secondo metodo. Teniamo abbassato il tasto ← e quindi premiamo REPT. Appena il cursore raggiunge l'errore alziamo il dito da REPT: abbiamo così cancellato tutti i caratteri su cui è passato il cursore. Se siamo andati troppo a sinistra battiamo allora → per rimemorizzare i caratteri corretti. Quando il cursore è sulla S battiamo una D così da avere la parola LOAD corretta. Ma che cosa succede se battiamo adesso RETURN?

```
LOAD▒COLOR DEMOSOFT
```

Tutti i caratteri a destra del cursore saranno cancellati e quindi l'istruzione che diamo diviene imprecisa:

```
LOAD
```

Prima di premere RETURN dovete riportare il cursore a destra e questo lo potete fare con il tasto ←.

RESET ACCIDENTALE

Se il vostro calcolatore non prevede il doppio comando CTRL-RESET per dare RESET, allora è probabile che premiate questo tasto inavvertitamente.

Per rendere più difficile la battitura di RESET vi diamo alcuni suggerimenti pratici. La cosa più semplice che potete fare è togliere la parte in plastica del tasto (vedi Figura 2.9) e lasciare solo la parte interna che è molto più difficile da premere.

È possibile anche inserire un anellino di gomma che renda invece più duro premere RESET. Prendete infatti un anellino con diametro esterno di circa 12 mm, diametro interno 9 mm e spessore 3 mm. applicatelo tra il tasto e la parte interna come indicato nella Figura 2.10. Rimettete poi il tasto sopra l'anello.

Ricupero dopo RESET

Il ricupero del vostro programma, se avete inavvertitamente premuto RESET, non sempre è possibile e dipende sia dalla particolare istruzione che in quel preciso momento veniva eseguita, sia dal tipo di Apple II che avete.

Ogni programma che usate dovrebbe indicarvi che cosa fare se premete RESET. Se per esempio state eseguendo un programma in BASIC *dovreste* sempre poterlo rieseguire dall'inizio. Ma questa è certamente una magra consolazione!

Quando premete RESET, il calcolatore ferma qualunque cosa stia facendo e passa il controllo alla tastiera. Un carattere di pronto e il cursore ritornano sullo schermo. Se il carattere di pronto è quello del BASIC, allora potete dare nuovamente il RUN e forse non tutto quello che avete fatto prima è andato perso.

Se invece vedete il carattere di pronto del Monitor (*), allora dovete passare al BASIC battendo CTRL-C *a meno che* non stiate lavorando con l'Applesoft su cassetta o su dischetto. Per passare dal Monitor all'Applesoft su disco battete il comando:

```
*3D0G
```

Per passare invece dal Monitor all'Applesoft su cassetta battete il comando:

```
*0G
```



Figura 2.9 – Un consiglio per non premere accidentalmente RESET: fase I

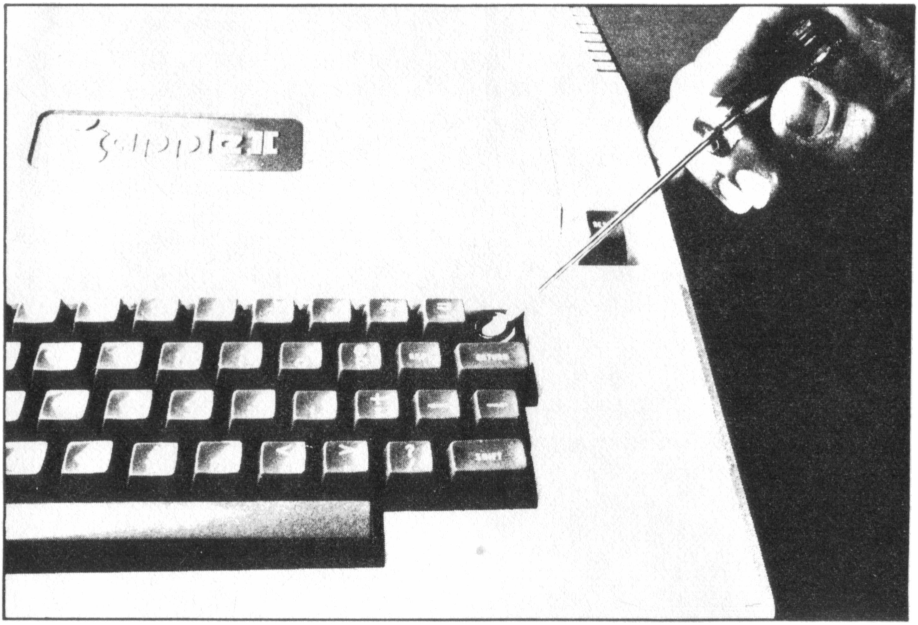


Figura 2.10 – Un consiglio per non premere accidentalmente RESET: fase II.

ATTENZIONE: Prima di battere questi comandi dovete essere ben sicuri di quello che fate! Se date un comando sbagliato rischiate di dover ricaricare tutto: DOS, Applesoft e il vostro programma.

CAPITOLO 3

PROGRAMMAZIONE IN BASIC

Questo capitolo è dedicato all'insegnamento del linguaggio di programmazione BASIC.

Il BASIC, come ogni altro linguaggio di programmazione, è composto da un insieme di istruzioni che opportunamente combinate permettono di risolvere un vostro specifico problema.

Potremmo iniziare la presentazione del BASIC descrivendo una per una tutte le sue istruzioni, ma ciò comporterebbe da parte vostra di imparare a memoria tante regole senza avere una chiara visione di che cosa significhi programmare. Per questo motivo abbiamo rimandato al capitolo 8 la definizione rigorosa delle istruzioni BASIC, mentre in questo capitolo vi diamo un approccio pratico alla programmazione. Il capitolo 8 può essere però consultato ogni volta che avete un dubbio sull'esatto significato di una istruzione.

AVVIO DELL'INTERPRETE BASIC

Nei calcolatori Apple II sono disponibili due diversi interpreti BASIC. Alcuni Apple II hanno solamente l'Integer BASIC, altri hanno l'Applesoft e altri ancora hanno ambedue questi linguaggi. Per il momento non è necessario farne una trattazione separata perchè sono molto analoghi; quando le differenze saranno rilevanti indicheremo espressamente di quale linguaggio si tratta.

Verifica del carattere di pronto BASIC.

I calcolatori Apple II sono multilingue per cui se volete lavorare in BASIC dovete controllare che il carattere di pronto, che vi appare sullo schermo, si riferisca ad un interprete BASIC.

Nel capitolo 2 abbiamo già illustrato come ottenere uno di questi caratteri di pronto; ripetiamo ora brevemente come procedere.

Se il vostro calcolatore ha 'Autostart Monitor è allora sufficiente accenderlo e premere RESET (oppure CTRL-RESET per alcune speciali tastiere). Diversamente dovete premere CTRL-B e RETURN. Vi ricordiamo che l'Integer BASIC è indicato dal simbolo (>) prima del cursore e l'Applesoft dal simbolo (]).

MODI DI PROGRAMMAZIONE IMMEDIATO E DIFFERITO

Iniziamo con alcune semplici istruzioni BASIC che possono essere eseguite sia con l'Integer BASIC che con l'Applesoft, cioè non preoccupatevi di quale carattere di pronto apparirà nei successivi esempi. Solo più avanti *vi daremo* indicazioni esplicite di quando una istruzione è valida solo per uno dei due interpreti.

VISUALIZZAZIONE DI CARATTERI

Appena il vostro Apple II inizia a lavorare in BASIC, esso è in *modo immediato* cioè può lavorare come una calcolatrice tascabile (tale modo viene detto anche diretto). Nel modo immediato il calcolatore esegue subito ogni istruzione che gli date. Per esempio battete:

```
PRINT "BUON GIORNO A TUTTI"
```

Appena premete RETURN vi apparirà sullo schermo il messaggio da voi posto tra virgolette:

```
BUON GIORNO A TUTTI
```

Se invece vi appare un avviso di errore come? SYNTAX ERROR oppure *** SYNTAX ERR, vuol dire che avete dato una istruzione errata. Forse avete dimenticato di scrivere PRINT. Se invece vi appare la cifra 0, significa che non avete posto le virgolette di sinistra.

Per correggere l'istruzione è sufficiente ribatterla. Fate però sempre molta attenzione a commettere meno errori possibile, specialmente nell'ortografia e nella punteggiatura, perché ciò può portare a risultati di elaborazione completamente errati!

Il comando che avete dato permette di visualizzare qualunque testo che avete posto tra virgolette.

Esiste però un limite massimo alla lunghezza dei testi che potete mettere tra virgolette e questo limite è diverso tra l'Integer BASIC e l'Applesoft. In ambedue i casi però un comando può essere più lungo di una linea del vostro schermo. Questo significa che un comando, tipo quello che stiamo ora esaminando, va automaticamente a capo (nella terminologia si dice "Wrap around"). Provate infatti a battere:

```
]PRINT "L'IMPORTANZA DI DARWIN NELLA BIO
LOGIA E' PARI A QUELLA DI GALILEO NELLA
FISICA"
L'IMPORTANZA DI DARWIN NELLA BIOLOGIA E'
 PARI A QUELLA DI GALILEO NELLA FISICA
```

Con l'Integer BASIC potete usare fino a 120 caratteri. Se superate questo limite avrete l'errore *** TOO LONG ERR dopo aver premuto RETURN.

Con l'Applesoft potete arrivare a 255 caratteri. Mentre vi avvicinate a questo limite il calcolatore comincia a fare "beep" e al superamento del limite appare una barra inversa (\) che vi cancella l'ingresso come se voi aveste dato il comando CTRL-X.

VISUALIZZAZIONE DI CALCOLI

Potete usare l'Apple II in modo immediato come una calcolatrice: ottenete cioè subito il risultato dei vostri calcoli. Provate con questi esempi:

```
PRINT 4+6                  Addizione
10

PRINT 500-437              Sottrazione
63

PRINT 100*23               Moltiplicazione
2300

PRINT 96/12                Divisione
8

PRINT 3^2                  Potenza
9

PRINT 3*4*10-800           Espressione
-680
```

Il risultato del vostro calcolo appare sempre sulla linea immediatamente successiva. Notate che in questi esempi non avete usato le virgolette. Provate a porle e vedere quindi che cosa succede.

L'Integer BASIC pone un limite massimo e un minimo al valore dei calcoli. Se il risultato di un calcolo, o di una operazione intermedia, è superiore a 32767 appare il messaggio di errore * * * > 32767 ERR. Se è invece inferiore a –32767 appare il messaggio di errore – * * * > 32767 ERR. Vi mostriamo alcuni esempi; in particolare nell'ultimo potete vedere che cosa succede se tentate di fare una divisione per zero.

```
>PRINT -32766-2
-*** >32767 ERR

>PRINT 2^15-1
*** >32767 ERR

>PRINT 10/0
*** >32767 ERR
```

L'Integer BASIC non prevede l'uso di numeri con parte frazionaria. Se quindi provate a fare una divisione che porterebbe ad un numero non intero, la parte frazionaria (o decimale) viene troncata via. Per esempio provate:

```
>PRINT 9/2
4
```

Il linguaggio Applesoft permette invece l'uso di numeri con parte frazionaria. In questo caso le cifre totali *significative* possono essere fino a nove comprese quelle intere e quelle frazionarie. Nel caso di numeri, con più di nove cifre significative, il calcolatore fa un arrotondamento a nove cifre. Ecco alcuni esempi:

```
]PRINT 12.34567896          APPROSSIMATO
12.345679                   PER ECCESSO

]PRINT 12.34567894          APPROSSIMATO
12.3456789                  PER DIFETTO
```

Se provate a fare degli altri esempi, può succedere che alcuni risultati vi appaiano sotto forma di *notazione scientifica:*

```
]PRINT 123456789123
1.23456789E+11
```

Se non conoscete questa forma di rappresentazione numerica, limitatevi per ora a fare dei calcoli molto semplici. Nel seguito di questo capitolo vi spiegheremo che cosa è la notazione scientifica di un numero.

Istruzione PRINT abbreviata

L'Applesoft permette di abbreviare l'istruzione PRINT con il punto interrogativo (?). Ecco alcuni esempi:

```
]?"IL TEMPO E' DENARO"
IL TEMPO E' DENARO

]?13-46*6
-263
```

MESSAGGI DI ERRORE

Abbiamo già incontrato alcune situazioni in cui il calcolatore Apple II rivela un errore di programmazione o di funzionamento. In questi casi il calcolatore suona un

"beep" per richiamare la vostra attenzione e nel contempo tenta di diagnosticare l'errore. Non pensiate però che un calcolatore possa riconoscere qualunque causa di errore; esso vi può dare al massimo una indicazione di quale genere di errore avete commesso. L'Apple II riconosce infatti 35 possibili tipi di errore ognuno dei quali può essere dovuto a moltissime diverse cause.

In questo capitolo, come anche nei successivi, vi illustreremo alcuni errori che si possono verificare in determinate situazioni concrete.

Nella Appendice C trovate invece la lista completa in ordine alfabetico di tutti i messaggi di errore.

Formato dei messaggi di errore

I messaggi di errore presentano un formato diverso tra l'Integer BASIC e l'Applesoft:

```
]PRNIT "IL SOLE SORGE"

?SYNTAX ERROR
]


PRNIT "IL SOLE SORGE"
*** SYNTAX ERR
>
```

SPAZIATURA NELLE ISTRUZIONI

Se vi siete posti il problema di dove potete, o non potete, inserire spazi vuoti in una istruzione BASIC, sappiate che ciò non ha molta importanza in quanto l'Apple II sa riconoscere il testo di una istruzione BASIC indipendentemente dagli spazi bianchi. L'unico caso in cui gli spazi bianchi vengono riconosciuti e mantenuti è quando sono battuti in un testo tra virgolette.

ISTRUZIONI, LINEE E PROGRAMMI

Un programma è un insieme di istruzioni che permettono ad un calcolatore di eseguire una ben determinata procedura di calcolo (o di effettuare una certa elaborazione di informazioni). Talvolta il problema in oggetto è semplice e breve, per cui anche il programma sarà corto. Tutte le istruzioni in modo immediato che vi abbiamo indicato come esempi, sono in effetti dei programmi. In questi casi il programma aveva una

sola istruzione; generalmente invece un programma può avere anche 10, 100, 1000 o più istruzioni. Consideriamo le seguenti istruzioni:

```
PRINT "EVVIVA"
EVVIVA

PRINT "CHI E' STATO?"
CHI E' STATO?

PRINT "FATTURA NO."
FATTURA NO.
```

Ognuno di questi programmi è costituito da una istruzione in modo immediato, posta su una unica linea e il risultato appare su un'altra linea.

L'Applesoft vi permette di porre più istruzioni su una stessa linea purchè le separiate con il carattere due punti (:). Le tre istruzioni precedenti possono infatti essere riscritte come un'unica istruzione:

```
]PRINT "EVVIVA":PRINT "CHI E' STATO?"
:PRINT "FATTURA NO."
EVVIVA
CHI E' STATO?
FATTURA NO.
```

e portano allo stesso risultato.

Su una linea di programma Applesoft potete inserire un numero anche cospicuo di istruzioni. L'unica limitazione deriva dal fatto che una linea in Applesoft non può essere più lunga di 255 caratteri. Se battete una linea molto lunga, il calcolatore vi suona un "beep" quando superate il 248-esimo carattere. Se superate poi il limite massimo di 255 caratteri, la linea viene totalmente cancellata come se aveste dato il comando CTRL-X. Se vi succede questo dovete ribattere tutta la linea dall'inizio. Comprendete quindi che esiste un limite fisico al numero di istruzioni che si possono porre su una stessa linea.

Programmi in Applesoft con una sola linea

Desideriamo darvi ora un esempio di che cosa si possa fare con una sola linea data in modo immediato. Battete per esempio la seguente linea;

```
]FOR I=1 TO 800:?"A";:NEXT :?"UFFA!"
```

Per il momento non chiedetevi il significato di questo "mini programma", ma battetelo esattamente come sopra indicato (alla fine battete anche RETURN). Sullo scher-

mo vedrete apparire una successione di 800 lettere A disposte su 20 righe e 40 colonne. Alla 21-esima riga vi appare anche l'esclamazione UFFA!

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
UFFA!
```

La linea di programma rimane sullo schermo perchè non avete superato il numero massimo di righe consentite.

Al termine del programma riappare il carattere di pronto dell'Applesoft seguito dal cursore.

MODO DIFFERITO

Lavorare in modo immediato non permette ovviamente di eseguire programmi che abbiano una certa consistenza. In Integer BASIC c'è anche la limitazione di potere scrivere una sola istruzione su una linea. Un'altra limitazione del mondo immediato è quella per cui dovete ribattere la linea ogni volta che volete eseguirla. In seguito vi mostreremo che potete rieseguire una linea purchè essa sia rimasta presente sullo schermo, ma ciò non comporta vantaggi veramente notevoli.

È possibile superare le limitazioni del modo immediato e cioè scrivere programmi più lunghi di una sola linea, memorizzarli ed eseguirli in un secondo tempo, con il *modo differito* (detto anche *a programma* o *indiretto*). Con il modo differito il calcolatore memorizza tutte le linee che battete senza eseguirle. Poi quando darete uno speciale comando il calcolatore eseguirà le istruzioni tutte assieme in successione.

Esecuzione di un programma

Si dice che un calcolatore *esegue* (in inglese *execute* oppure *run*) un programma quando esso compie le operazioni indicate nel programma stesso.

In modo immediato un programma viene eseguito appena premete il tasto RETURN.

In modo differito un programma viene invece eseguito quando date il comando RUN. Ogni volta che ripetete questo comando il programma viene rieseguito nuovamente.

Come cancellare i vecchi programmi

Dal momento che un programma scritto in modo differito rimane nella memoria del calcolatore, prima di batterne uno nuovo dovete cancellare quello vecchio.

Per ottenere questo è sufficiente che battiate il comando NEW all'inizio del nuovo programma. Se vi dimenticate di farlo il nuovo programma risulterà sovrapposto a quello vecchio.

Come indicare la fine di un programma

La fine di un programma in modo immediato è ovvia.

Nel caso invece di programmi in modo differito è necessario porre una istruzione END per fermare la sua esecuzione far tornare al modo immediato il calcolatore. È chiaro che la END deve essere l'ultima istruzione logica di un programma.

L'Applesoft non richiede che si ponga l'istruzione END, esso termina appena non trova più istruzioni.

Numeri di linea

I numeri di linea permettono di definire un programma in modo differito. Un numero di linea è un numero intero sino a cinque cifre posto subito all'inizio della linea. La presenza, oppure no, di un numero all'inizio di una linea permette di distinguere tra istruzioni in modo differito e quelle in modo immediato.

In seguito vedremo alcune istruzioni che possono essere eseguite solo in uno dei due modi e non nell'altro.

Proviamo un primo esempio di programma in modo differito:

```
>NEW

>10 PRINT "RE DI PICCHE"
>20 END
>RUN
RE DI PICCHE
```

Ogni numero di linea deve essere unico. Se battete più linee con lo stesso numero, solo l'ultima verrà eseguita nel programma. Provate infatti a battere il seguente esempio con due linee numero 10:

```
>NEW

>10 PRINT "LINEA 10"
>10 PRINT "LINEA = 10"
>20 END
>RUN
LINEA = 10
```

I numeri di linea hanno anche lo scopo fondamentale di stabilire l'ordine con cui saranno eseguite le singole linee. La prima linea logica deve avere il numero più basso, mentre l'ultima quello più alto. È possibile battere le linee di programma con i numeri non in sequenza ordinata. Il calcolatore Apple II provvederà automaticamente a porle in ordine crescente. Consideriamo per esempio questo programma scritto con le linee non ordinate:

```
>NEW

>30 PRINT "CASA"
>10 PRINT "ALBERO"
>20 PRINT "GATTO"
>40 PRINT "XXX"
>50 END
>RUN
ALBERO
GATTO
CASA
XXX
```

Per dimostrarvi che il calcolatore non perde un programma differito, pulite lo schermo con il comando ESC-@ e date nuovamente RUN:

```
>             PREMERE ESC-@ E RETURN

>RUN
ALBERO
GATTO
CASA
XXX
```

Se volete potete sempre aggiungere nuove linee ad un programma memorizzato. Le nuove linee possono essere poste all'inizio, in coda oppure entro il programma. Per fare questo è sufficiente dare un nuovo numero di linea. Supponiamo di voler aggiungere una istruzione all'inizio del nostro esempio; basta battere la nuova linea con

un numero inferiore a 10. Ricordatevi però di non dare il comando NEW perchè cancellereste il programma! Supponiamo che la nuova linea abbia il numero 5:

```
>5 PRINT "ECCETERA"
>RUN
ECCETERA
ALBERO
GATTO
CASA
XXX
```

Notate che se il programma originale fosse iniziato con il numero 0 non sarebbe stato possibile anteporre nessuna nuova linea. È consigliabile infatti scrivere un programma con un numero iniziale abbastanza alto e lasciare dei numeri liberi tra una linea e la successiva.

Linee di programma con più istruzioni

È possibile porre più istruzioni su una unica linea. La prima deve seguire il numero di linea mentre le altre seguono in successione logica separate da due punti (:).

Anche in Integer BASIC è possibile porre più istruzioni su una stessa linea, quando si lavora in modo differito, a differenza di quanto è consentito invece per il modo immediato. La lunghezza della linea è approssimativamente di 150 caratteri, ma ciò dipende dal contenuto della linea stessa. Se superate i limiti di una riga il calcolatore vi avvisa con il messaggio di errore * * * TOO LONG ERR. In tal caso dovete ribattere la linea.

Con l'Applesoft è possibile invece porre più istruzioni su una stessa linea sia in modo immediato che differito. In ambedue i casi una linea non può superare i 255 caratteri come abbiamo già indicato.

Listato di un programma

Con il comando LIST potete vedere in ogni momento quale programma è contenuto nella memoria del calcolatore. Provate subito purchè non abbiate cancellato il programma con NEW. Sullo schermo dovrebbero apparirvi le seguenti linee di programma:

```
LIST

5 PRINT "ECCETERA"
10 PRINT "ALBERO"
20 PRINT "GATTA"
30 PRINT "CASA"
40 PRINT "XXX"
50 END
```

Questo è chiamato un *listato di programma*. È possibile anche listare una sola linea o un gruppo di linee. Questa seconda possibilità è molto utile quando avete un programma molto lungo che non può apparire tutto insieme sullo schermo. Proviamo allora a listare una linea del nostro programma:

```
LIST 10
```

Sullo schermo appare:

```
10 PRINT "ALBERO"
```

Per listare invece un gruppo di linee dovete indicare la prima e l'ultima del gruppo:

```
LIST 20,40

20 PRINT "GATTO"
30 PRINT "CASA"
40 PRINT "XXX"
```

In Applesoft è possibile listare un programma dall'inizio sino ad una certa linea, come anche da una linea sino alla fine. Ecco alcuni esempi:

```
]LIST ,10

5 PRINT "ECCETERA"
10 PRINT "ALBERO"


]LIST 30,

30 PRINT "CASA"
40 PRINT "XXX"
50 END
```

Interruzione di un listato

Se volete fermare la visualizzazione di un listato, prima che raggiunga la fine, dovete dare il comando CTRL-C. Questa possibilità è molto utile se, mentre state listando un programma molto lungo, vi accorgete che non vi interessa più e volete fermarlo.

Se il vostro calcolatore ha l'Autostart Monitor, potete fermare temporaneamente la visualizzazione del listato premendo CTRL-S. Per farla proseguire basta poi premere il tasto (o barra) di spaziatura. CTRL-S vi permette in pratica di leggere a blocchi un lungo programma.

Numerazione automatica delle linee

Con l'istruzione AUTO è possibile numerare automaticamente le linee di un programma scritto in Integer BASIC. Ogni volta che premete RETURN, il calcolatore pone il nuovo numero di linea purchè nella linea precedente non vi siano errori oppure essa sia vuota.

Per uscire dalla numerazione automatica battete CTRL-X. Esso cancella il nuovo numero di linea. Subito dopo battete MAN per ritornare alla numerazione manuale.

Per vedere in pratica come funzionano i due comandi AUTO e MAN battete il seguente esempio:

```
>AUTO 1000

>1000 PRINT "QUANTE YARD IN UN MIGLIO?"

                               PREMERE RETURN
>1010
>1010 PRIMT 5280/3
*** SYNTAX ERR
>1010 PRINT 5280/3
>1020
                               PREMERE CTRL-X
>MAN
```

Notate che nell'istruzione AUTO dovete indicare il numero di linea da cui iniziare la numerazione. È possibile indicare anche il passo di numerazione. Ecco un esempio:

```
>AUTO 1000,100

>1000 PRINT "PIRIPICCHIO"
>1100
                               PREMERE CTRL-X
>MAN
```

Come vedete in questo esempio si è scelto di incrementare i numeri di linea di 100. Se non indicate alcun valore il passo viene posto implicitamente eguale a 10.

Il linguaggio Applesoft non prevede la numerazione automatica.

REGISTRAZIONE DI PROGRAMMI SU CASSETTA

Se avete collegato un registratore a cassette al vostro calcolatore, potete allora registrarvi sopra i vostri programmi, purchè scritti in modo differito, e poi riutilizzarli

quando volete. Supponiamo che abbiate questo programma già caricato nella memoria del calcolatore:

```
10 PRINT "NON"
20 PRINT "CREDO"
30 PRINT "DI"
40 PRINT "AVERLO"
50 PRINT "MAI"
60 PRINT "VISTO"
70 END
```

Per "salvare" questo programma (cioè registrarlo su una memoria esterna), ponete allora una cassetta nel registratore e riavvolgetela completamente. Premete quindi contemporaneamente i tasti RECORD e PLAY e battete subito alla tastiera l'istruzione:

```
SAVE
```

Il calcolatore suona un "beep" per avvisarvi che inizia la registrazione del programma e suonerà un'altro "beep" per indicare il termine della registrazione. Dopo il secondo "beep" premete subito il tasto STOP del registratore.

Per fare un controllo che la registrazione sia avvenuta effettivamente, battete i comandi NEW e LIST sia per cancellare la memoria che per controllare che essa non contenga più alcun programma.

Per caricare il programma dalla cassetta per prima cosa dovete riavvolgerla, poi premere il tasto PLAY del registratore e battere l'istruzione:

```
LOAD
```

Anche questa volta il calcolatore suona due volte "beep" per indicare l'inizio e la fine della lettura del programma. Subito dopo il secondo "beep" fermate il registratore premendo il suo tasto STOP. Per controllare che il programma è ora presente nella memoria del calcolatore, date il comando LIST.

Nel capitolo 5 vi indicheremo come registrare e leggere programmi su dischetto. L'uso dei dischetti è senz'altro più conveniente delle cassette.

Registrazione di più programmi su una cassetta

Avrete sicuramente notato che un programma occupa poco spazio su una cassetta. Ovviamente più un programma è lungo, più nastro esso occupa, ma ben difficilmente un programma può impegnare tutta una cassetta per cui è possibile registrare su una di esse sequenzialmente più programmi.

Caricare però nel calcolatore i programmi successivi al primo non è cosi semplice come caricare il primo programma. Generalmente dovete riavvolgere tutta la casset-

ta e caricare uno per volta tutti i programmi sino a quello che vi interessa. Cioè dovete dare più volte il comando LOAD e ripetere la procedura più sopra descritta.

Se il vostro registratore ha invece un contagiri, potete allora segnarvi a parte a quale numero di giri corrisponde un programma e quindi accedere direttamente a quella porzione di nastro.

Il primo programma inizia ovviamente con il contatore a zero. Il secondo programma inizia invece con il numero finale dei giri del primo programma. E così via: ad ogni programma associate il numero di giri finale del programma precedente.

Per caricare allora un programma da un punto qualunque della cassetta, fate avanzare il nastro con il tasto FAST FORWARD sino al numero di giri che gli compete (se andate troppo in avanti potete tornare in dietro con REWIND) poi iniziate la normale procedura battendo LOAD.

CAMBIO DELLA VERSIONE DI BASIC

Molti calcolatori Apple II vi permettono di scegliere tra l'Integer BASIC e l'Applesoft. I motivi per usare una versione o l'altra vi saranno più evidenti proseguendo la lettura di questo capitolo. Come esempio vi possiamo ricordare che l'Integer BASIC permette la numerazione automatica delle linee, mentre ciò non è possibile con l'Applesoft. Nello stesso tempo abbiamo già visto che l'Applesoft permette l'uso di numeri frazionari e l'Integer BASIC solo di numeri interi.

Mentre molti calcolatori Apple II hanno ambedue le versioni di BASIC, alcuni non le hanno come per esempio l'Apple II Plus che ha solo l'Applesoft. La procedura per cambiare versione di BASIC dipende poi dal modello di calcolatore Apple II e dalle opzioni che vi sono installate.

Con l'opzione "Apple Language System" potete accedere immediatamente ad ambedue le versioni di BASIC. Se siete in Integer BASIC battete FP per passare all'Applesoft. Diversamente battete INT per passare dall'Applesoft all'Integer BASIC.

Anche la cartolina con il firmware Applesoft permette l'accesso immediato alle due versioni di BASIC. Su questa cartolina vi è un interruttore che è accessibile dal retro del calcolatore. Se questo interruttore viene portato nella posizione bassa, prima di accendere il calcolatore, quando poi lo accenderete otterrete l'Integer BASIC. Se invece alzate tale interruttore, sempre con il calcolatore spento, otterrete all'accensione l'Applesoft.

Sempre se la cartolina firmware è installata potete invece battere FP per avere l'Applesoft e INT per avere l'Integer BASIC.

Su un calcolatore Apple II standard è molto facile accedere all'Integer BASIC. Premete RESET e battete CTRL-B. Per avere invece l'Applesoft la situazione è più complessa perchè esso deve essere caricato in memoria da dischetto o da cassetta.

In quest'ultimo caso, e se avete l'Applesoft su dischetto, dovete prima caricare il DOS come descritto nel capitolo 2. Ripetiamo qui brevemente la procedura che deve

essere eseguita ogni volta che accendete il calcolatore. Per caricare quindi il DOS, a partire dal Monitor o dall'Integer BASIC, battete il comando:

```
*6                    PREMERE CTRL-P E RETURN

>PR#6
```

Quando il DOS è stato caricato potete richiamare l'Applesoft dall'Integer BASIC battendo il comando FP (ricordatevi che nel drive sia presente il dischetto con l'Applesoft). Pochi secondi dopo vi apparirà sullo schermo il carattere di pronto dell'Applesoft (]).

Se il vostro Applesoft è registrato invece su cassetta, dovete seguire la procedura indicata nel capitolo 2. Brevemente vi ricordiamo che dovete dapprima inserire la cassetta nel registratore, poi premere il tasto PLAY e quindi dare, alla tastiera, il comando LOAD in Integer BASIC. Dopo circa 1 o 2 minuti sullo schermo vi apparirà il carattere di pronto dell'Applesoft.

Dall'Applesoft potete ritornare all'Integer BASIC battendo il comando INT. Se volete passare ancora all'Applesoft battete il comando FP nel caso del dischetto e LOAD nel caso della cassetta.

TECNICHE AVANZATE DI EDITING

Nel capitolo 2 abbiamo visto alcune possibili tecniche per correggere una linea che state battendo, prima di premere RETURN. Rivediamole brevemente:

Con il tasto ← spostate all'indietro il cursore cancellando i caratteri dalla memoria (sebbene rimangano presenti sul video).

Con il tasto → spostate in avanti il cursore ricopiando in memoria i caratteri su cui passa sopra.

Il tasto REPT, usato assieme a ← e a →, permette le stesse operazioni in minor tempo.

CTRL-X cancella la linea che state battendo.

ESC-@ pulisce tutto lo schermo e porta il cursore in alto a sinistra.

Vediamo ora nuove tecniche di editing che sono particolarmente utili per correggere o modificare il testo di un programma dato in modo differito (cioè con i numeri di linea).

CANCELLAZIONE DI LINEE DI PROGRAMMA

Per cancellare una linea intera battete il suo numero di linea e subito dopo RETURN. Se provate poi a fare un listato vi accorgerete che ne la linea, ne il suo numero sono presenti. Ecco un esempio:

```
>100 PRINT "ROSSI PIETRO"
>110 PRINT "LEONE CARLO"
>120 PRINT "FABBRI ENZO"
>130 PRINT "NATOLI MARIO"
>140 PRINT "PORRO LUIGI"
>150 END

>110
>130

>LIST

100 PRINT "ROSSI PIETRO"
120 PRINT "FABBRI ENZO"
140 PRINT "PORRO LUIGI"
150 END
```

Se dovete cancellare un intero blocco di linee potete usare il comando DEL. Per esempio:

```
>DEL 110,140

>LIST

100 PRINT "ROSSI PIETRO"
150 END
```

Il comando DEL 110,140 cancella tutte le linee dalla 110 alla 140 anche se la 110 non esisteva.

AGGIUNTA DI NUOVE LINEE DI PROGRAMMA

Potete in qualunque momento battere nuove linee di programma con qualunque ordine. La loro posizione nel programma dipenderà dal numero di linea. Il calcolatore fa infatti un inserimento automatico delle nuove linee a seconda della loro numerazione. Provate ora ad aggiungere le linee 110 e 120 nell'esempio di prima:

```
>120 PRINT "FABBRI ENZO"
>110 PRINT "LEONE CARLO"

>LIST

100 PRINT "ROSSI PIETRO"
110 PRINT "LEONE CARLO"
120 PRINT "FABBRI ENZO"
150 END
```

CAMBIAMENTO DI LINEE DI PROGRAMMA

Il modo più semplice per cambiare una linea di programma è quello di ribatterla. Ma questo metodo è molto precario sia perchè può richiedere molto tempo per ribattere tutta una linea, sia perchè ogni volta che si batte qualcosa di nuovo aumenta la probabilità di commettere nuovi errori.

Fortunatamente esiste una possibilità di riusare, ciò che è già stato battuto, sia in Integer BASIC che in Applesoft. Questa possibilità è legata al fatto che il testo che appare sullo schermo è da considerarsi *vivo*. Potete così fare dell'editing di tutto ciò che è sullo schermo. Mediante il tasto ESC, assieme ad altri tasti, potete muovere il cursore sullo schermo a vostro piacimento. Potete allora portare il cursore su qualunque linea presente sullo schermo, ricopiare le parti che vi interessano con il tasto → e quindi cancellare, inserire o sostituire caratteri ovunque su una linea.

Vediamo ora come questo metodo funziona in pratica.

Per prima cosa date il comando LIST per visualizzare il testo che vi interessa. Notate subito che ottenete una maggiore spaziatura per rendere il testo più facilmente leggibile. (Ciò può essere un inconveniente per programmi molto lunghi!) Se volete un listato senza questa maggiore spaziatura, date allora il comando per pulire lo schermo ESC-@ e poi battete:

```
POKE 33,33
```

Questo comando, oltre a togliere la spaziatura aggiuntiva, riduce anche la larghezza dello schermo da 40 a 33 colonne. Il comando POKE sarà descritto nei dettagli nel capitolo 4. Per riottenere lo schermo normale battete invece:

```
POKE 33,40
```

Movimenti del cursore

Per promuovere il cursore in qualunque punto dello schermo dovete premere in successione ESC e una delle lettere A, B, C o D. ESC deve essere battuto ogni volta che battete una delle quattro lettere per muovere di una posizione il cursore a destra, a sinistra, in alto o in basso come indicato nella Figura 3.1.

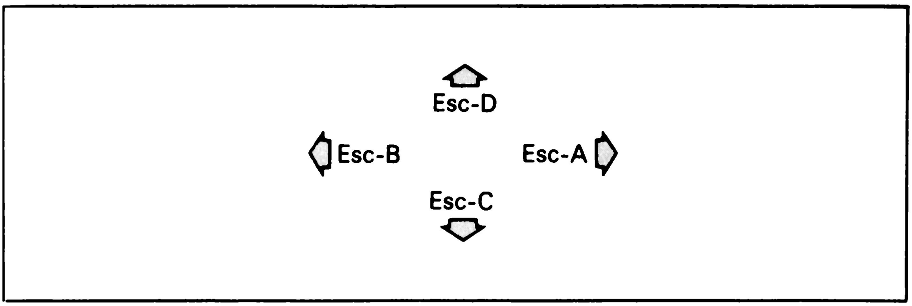


Figura 3.1 – Movimenti del cursore (Sequenza con due tasti)

Se avete l'Autostart Monitor potete anche usare le lettere I, J, K e M dopo ESC per muovere il cursore. Questa soluzione presenta un vantaggio pratico perchè le quattro lettere sono poste vicine sulla tastiera (vedere la Figura 3.2).

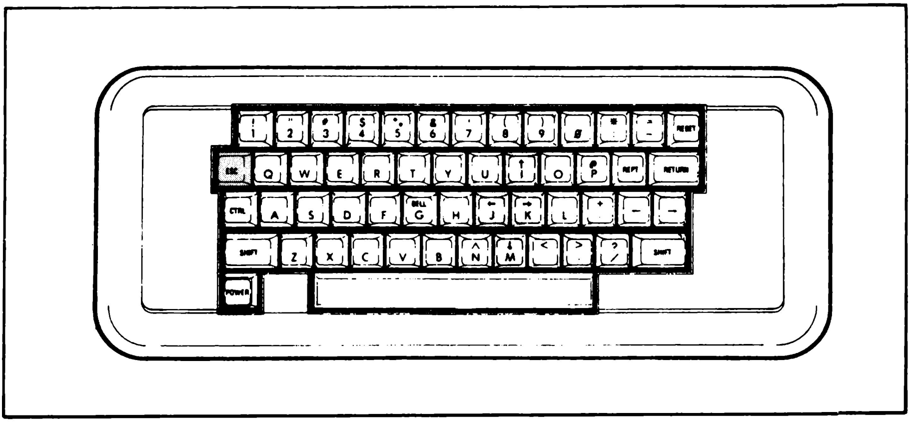


Figura 3.2 – Movimenti del cursore (Versione con l'Autostart Monitor)

Il vantaggio di questa seconda soluzione è anche un'altro e riguarda il funzionamento del tasto ESC. Se infatti premete ESC il calcolatore va in "edit mode" e potete quindi premere i tasti I, J, K e M quante volte volete per spostare il cursore senza dover ribattere ESC. Per rimanere in "edit mode" potete battere solo i tasti I, J, K, M, REPT, CTRL e SHIFT. Se battete un qualunque altro tasto uscite dall'"edit mode". L'uso di REPT è ovviamente quello di ripetere automaticamente un certo comando.

Cambiamento di caratteri

È molto semplice cambiare alcuni caratteri. Basta portare il cursore sopra al vecchio carattere e battere quello nuovo nella stessa posizione. Per esempio se avete il cursore come indicato qui di seguito:

```
100 PRINT "ORARIO DI ARRIVO"
```

potete cambiare il testo in questo modo:

```
100 PRINT "ORARIO DI PARTENZA"
```

È molto importante però ricordare che solo premendo RETURN il cambiamento diventa effettivo in memoria.

Cancellatura di caratteri

Il modo più semplice per cancellare dei caratteri è quello di battervi sopra degli spazi vuoti. Ricordatevi che in BASIC gli spazi vuoti non hanno alcun valore a meno

che siano posti in un testo tra virgolette. Per muovere il cursore verso destra potete usare ESC-A oppure ESC-K in "edit mode". A differenza del tasto → le due sequenze ESC-A e ESC-K non ricopiano i caratteri su cui passano sopra. Se i caratteri che volete cancellare sono all'interno di due virgolette, è più facile usare ESC-A, oppure ESC-K in "edit mode", per passare sopra ai caratteri non desiderati.

Per cancellare tutti i caratteri dalla posizione del cursore sino alla fine della linea sul display, potete dare il comando ESC-E. Ciò equivale a premere la barra di spaziatura ripetutamente sino alla fine della linea senza però muovere il cursore. I caratteri sulla riga successiva del display non vengono cancellati anche se fanno parte della stessa linea di programma. Per esempio se premete ESC e E in questo esempio:

```
100 PRINT "E' MEGLIO AVERE▒AMATO E PERSO
 CHE NON AVERE MAI AMATO"
```

ecco ciò che succede:

```
100 PRINT "E' MEGLIO AVERE▒
 CHE NON AVERE MAI AMATO"
```

Attenzione! Se premete ora RETURN la linea 100 viene accettata solo fino alla posizione dove è adesso il cursore. Per ricopiarla interamente dovete passarvi sopra il cursore con il tasto →.

Con la sequenza ESC-F cancellate invece tutto lo schermo dalla posizione del cursore sino alla fine.

Inserzione di caratteri

La procedura per inserire caratteri in un testo è molto semplice anche se può sembrare inizialmente complessa. È importante precisare subito che il calcolatore Apple II non può "spingere" a destra i caratteri vecchi per far posto a quelli nuovi. In Figura 3.3 abbiamo schematizzato il principio con cui l'Apple II fa l'inserimento di caratteri. Questo è possibile grazie ai tasti che spostano il cursore e ricordando il fatto che, ciò che si vede sullo schermo, non è una esatta replica di ciò che è memorizzato.

Con l'esempio che segue cerchiamo di spiegarvi la procedura di inserimento. Anche se le istruzioni sono date in Integer BASIC, la procedura vale anche per l'Applesoft. Supponiamo di avere questa linea di programma:

```
>NEW

>10 PRINT "SUL DI RIALTO"
>
```

PONTE

10 PRINT "SUL DI RIALTO"

Figura 3.3 — Inserimento di caratteri

Per inserire la parola PONTE, cominciamo con il listare il programma:

```
>LIST

10 PRINT "SUL DI RIALTO"

>
```

Usando *solamente* i tasti per spostare il cursore (ESC, ecc.) portate il cursore all'inizio della linea:

```
>LIST

10 PRINT "SUL DI RIALTO"

>
```

Mediante il tasto → ricopiate tutta la prima parte della linea e fermate il cursore sulla D.

```
>LIST

10 PRINT "SUL DI RIALTO"

>
```

Portate il cursore di una riga in alto con ESC-D. Se su questa riga vi sono già dei caratteri, cancellateli con ESC-E.

```
>LIST
                ▒
10 PRINT "SUL DI RIALTO"

>
```

Battete la parola PONTE e uno spazio:

```
>LIST
                PONTE ▒
10 PRINT "SUL DI RIALTO"

>
```

Mediante i *soli* tasti di movimento del cursore portate il cursore a sinistra sopra la P. *Non* usate il tasto ← perchè questo cancellerebbe dalla memoria la parola PONTE!

```
>LIST
                  PONTE
10 PRINT "SUL DI RIALTO"

>
```

Premete ora ESC-C per riportare il cursore nella posizione originaria:

```
>LIST
                  PONTE
10 PRINT "SUL DI RIALTO"

>
```

A questo punto con il tasto → ricopiate l'ultima parte della linea e premete infine RETURN.

Se ora provate a listare la linea vedrete che essa contiene correttamente la parola PONTE:

```
>LIST
                  PONTE
10 PRINT "SUL DI RIALTO"

>LIST

10 PRINT "SUL PONTE DI RIALTO"

>
```

L'Appendice B contiene un riassunto dei comandi di editing.

RIESECUZIONE IN MODO IMMEDIATO

Per il fatto che tutto ciò che è presente sullo schermo è "vivo", è possibile rieseguire una istruzione in modo immediato purchè essa sia ancora visibile sullo schermo.

Se necessario, prima di rieseguirla, potete modificarla. In ogni caso prima di tutto portate il cursore all'inizio della linea. Questa operazione deve essere eseguita con i tasti ESC e A, B, C, D (oppure con i tasti ESC e I, J, K, M se avete l'Autostart Monitor). Ricopiate quindi l'istruzione con il tasto → ed eventualmente modificatela con le tecniche che vi abbiamo più sopra illustrato.

Facciamo un esempio. Scriviamo in modo immediato la formula per calcolare il volume di un parallelepipedo con dimensioni 10x25x8 cm:

```
>PRINT "VOLUME = ";10*25*8
VOLUME = 2000
```

Se ora desiderate calcolare il volume di un parallelepipedo con dimensioni diverse (per esempio 10 x 25 x 14 cm) portate il cursore all'inizio della linea. Battete poi il tasto → (o aiutatevi con REPT) per ricopiare l'istruzione sino al numero 8 che deve essere cambiato. Se per errore superate l'8 allora tornate in dietro con ←. Quando il cursore è sull'8 battete il nuovo valore 14 e premete infine RETURN.

```
>PRINT "VOLUME = ";10*25*14
VOLUME = 3500
```

LINGUAGGI DI PROGRAMMAZIONE

Un linguaggio di programmazione è un *mezzo di comunicazione* tra l'uomo e il calcolatore. Alcuni linguaggi sono di uso generale, come il BASIC, altri sono stati realizzati per scopi particolari cioè per applicazioni commerciali, scientifiche, grafiche, elaborazione di testi, ecc.

Altri linguaggi per uso generale come il BASIC sono il Pascal, il FORTRAN, il COBOL, l'APL, il PL/M, il PL-1 e il FORTH.

I calcolatori Apple II possono lavorare con vari linguaggi e tra questi il BASIC e il Pascal. Questo libro è dedicato unicamente alla programmazione in BASIC.

Nell'ambito di qualunque linguaggio di programmazione, le sue singole istruzioni devono essere scritte in base a delle regole precise e ben definite. L'insieme di queste regole costituiscono la *sintassi* di quel linguaggio. Ogni linguaggio di programmazione è definito quindi in modo univoco dalla sua sintassi.

I linguaggi di programmazione, come le lingue parlate, hanno i loro *dialetti.* Questi dialetti si distinguono per delle piccole varianti nella sintassi. Il calcolatore Apple II ha due dialetti del BASIC: l'Integer BASIC e l'Applesoft. A causa delle diversità nelle sintassi dei due dialetti BASIC, un programma scritto per uno dei due non può normalmente essere interpretato dall'altro dialetto. A maggior ragione un programma scritto per il calcolatore Apple II non può girare su un altro calcolatore anche se quest'ultimo usa un linguaggio BASIC.

Sappiate comunque che dopo aver imparato un linguaggio BASIC è molto facile imparare tutte le altre versioni.

Se ora analizziamo le regole della sintassi vediamo che alcune sono evidenti e non richiedono alcuna spiegazione. Le regole per l'addizione e la sottrazione, che abbiamo già incontrato, sono infatti ovvie. Molte altre regole non hanno invece alcun significato esplicativo e devono essere accettate cosi come sono proprio perchè sono

completamente arbitrarie. Per esempio perchè viene usato l'asterisco per indicare la moltiplicazione? Noi siamo abituati a usare la X, ma in tal caso il calcolatore non potrebbe distinguere tra il segno di prodotto e la lettera X. Di conseguenza è stato scelto un altro carattere e questo è l'asterisco (esso è usato da tutti i linguaggi di programmazione). Per la divisione si è scelta la barra / perchè sulle tastiere dei calcolatori non era presente l'altro segno già molto usato per la divisione (÷).

ELEMENTI DEL BASIC

La maggior parte delle regole della sintassi BASIC riguarda le singole istruzioni. Queste regole riguardano tre componenti fondamentali di ogni istruzione: il numero di linea, le istruzioni al calcolatore e i dati. Esistono poi alcune regole pertinenti al programma nel suo insieme che descriveremo nel seguito di questo capitolo.

NUMERAZIONE DELLE LINEE

Abbiamo già visto che le linee date in modo differito devono essere numerate. Tali numeri devono essere unici e dati in ordine crescente.

In Integer BASIC il numero di linea deve essere intero, può avere sino a cinque cifre ed essere compreso tra 0 e 32767.

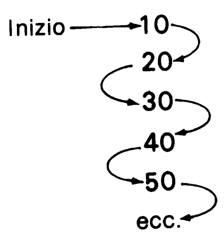
In Applesoft il numero di linea deve essere egualmente intero, con cinque cifre al massimo, ma può andare da 0 a 63999.

Numeri di linea e indirizzi

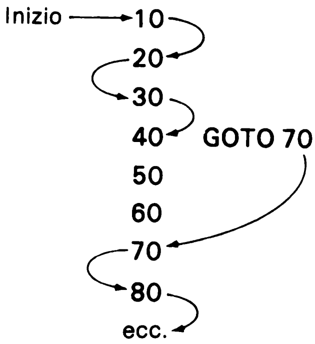
I numeri di linea equivalgono a indirizzi delle linee di programma. Questo concetto è molto importante poichè ogni programma può contenere due tipi di istruzioni:

1. Istruzioni che creano o modificano dati.
2. Istruzioni che controllano la sequenza con cui si devono svolgere le operazioni.

Il concetto che le operazioni indicate in un programma debbano essere eseguite con un ben definito ordine, è senz'altro facile e comprensibile. Normalmente l'esecuzione di un programma inizia dalla prima istruzione e prosegue verso l'ultima:



In seguito vedremo che spesso è necessario eseguire dei salti, per cui la numerazione delle linee diviene molto importante. Nell'esempio che segue, l'istruzione GOTO 70 in italiano significa VAI A 70:



SPAZI VUOTI

L'inserimento di spazi vuoti può essere fatto ovunque nelle istruzioni, ma ha unicamente lo scopo di rendere più leggibile il vostro programma. Dal momento però che tali spazi occupano memoria nel calcolatore, l'Apple II comprime gli spazi non necessari quando premete RETURN. Quando poi visualizzate il programma, con il comando LIST, l'Apple II reinserisce una opportuna spaziatura in base ad una predeterminata regola. È possibile sopprimere questo reinserimento mediante il comando POKE 33,33 battuto prima di quello LIST. Con POKE 33,40 ritornate alla situazione normale.

Ricordatevi però che gli spazi inseriti in un testo tra virgolette vengono sempre memorizzati. Guardate per esempio la differenza tra queste due istruzioni:

```
PRINT "DATA DI FATTURA"
DATA DI FATTURA


PR INT "    DATA DI.F ATTURA"
    DATA DI F ATTURA
```

DATI

Il compito principale di qualunque programma è quello di caricare, elaborare e fornire dati sia che essi siano numeri o testi. Il modo quindi con cui vengono trattati i dati è molto importante. Vediamo adesso quali sono i tipi di dati che si possono incontrare nel BASIC dell'Apple II.

Stringhe

Stringa è una qualunque sequenza di caratteri racchiusa tra virgolette. Abbiamo già usato le stringhe con l'istruzione PRINT per visualizzare dei messaggi. Ecco altri esempi di stringhe:

"CARLO ROSSI"
"ACCONTO 456-0067"
"18 MARZO 1956"

Salvo poche eccezioni, in una stringa potete inserire qualunque carattere della tastiera, alfabetico, numerico o di interpunzione, contemporaneo, o no, a CTRL e SHIFT.

Le eccezioni riguardano i movimenti del cursore e i comandi di termine delle linee. Essi sono ←, →, RETURN, ESC, CTRL-H, CTRL-M, CTRL-U e CTRL-X.

Le stringhe possono avere lunghezza da 0 a 255 caratteri. Una stringa senza caratteri è detta *stringa nulla*.

Premendo una certa combinazione di tasti si possono avere alcuni caratteri non visibili sullo schermo. Per esempio se premete CTRL-G ottenete un "beep". Questo suono può essere inserito in una stringa appunto con CTRL-G:

```
PRINT ""          PREMERE VARIE VOLTE
                  CTRL-G TRA LE VIRGOLETTE
```

In questo esempio potete udire il "beep" anche se non potete vederlo. Vi sono altri caratteri che non potete nè udire nè vedere. Essi in genere riguardano le funzioni di stampa, il controllo di dispositivi di comunicazione e altre cose che potete collegare all'Apple II.

Nell'Appendice I trovate l'insieme di tutti i caratteri dell'Apple II e come sono ottenibili dalla tastiera.

In Applesoft è possibile anche inserire dei caratteri in una stringa senza far uso dei tasti. Questo è ottenuto con la funzione CHR$ che descriveremo nel capitolo 4.

Numeri

L'Apple II prevede l'uso di due tipi di numeri: gli *interi* e i *numeri reali.* Gli interi sono numeri senza parte frazionaria; i reali (detti anche frazionari o con virgola mobile) sono quelli che hanno invece anche una parte frazionaria.

Come è implicito nel nome, l'Integer BASIC tratta solo numeri interi. L'Applesoft può trattare invece sia gli interi che i reali.

È importante precisare che la parte frazionaria dei numeri reali deve essere preceduta da un punto (come è nella abitudine anglosassone) e non da una virgola. I punti per separare le migliaia dai milioni, ecc. non devono mai essere posti! Per esempio dovete scrivere 6.5 e non 6,5 e inoltre dovete scrivere 32000 e non 32.000.

Interi

Un intero è un numero senza parte frazionaria. Esso può essere negativo (—) o positivo (+). Un numero senza segno è considerato positivo. I numeri interi possono essere compresi tra —32767 e +32767. Ecco alcuni esempi di interi:

0
1
44
3456
—59

Numeri reali

Un numero reale è caratterizzato dall'avere una parte frazionaria, ma può anche non averla. Può essere negativo (—) oppure positivo (+ o senza segno). Il numero reale più piccolo (cioè più negativo) è:

—100000000000000000000000000000000000000000

mentre quello più grande è:

100000000000000000000000000000000000000000

Ecco altri esempi di numeri reali:

5
—150
0
0.5
0.00654
1.37654
—546.87

Quando un numero reale è abbastanza vicino allo zero come per esempio 00000000000000000000000000000000000000003 allora esso sarà convertito in zero.

Notazione scientifica

In Applesoft i numeri reali molto grandi oppure molto piccoli sono rappresentati con la notazione scientifica. Ogni numero che abbia più di nove cifre davanti al punto decimale sarà tradotto nella notazione scientifica. Ogni numero più vicino allo zero di ±. 01 sarà convertito nella notazione scientifica.

Un numero nella notazione scientifica ha il formato:

numero E ± ee

dove:	*numero*	può essere un intero o un numero frazionario. Esso contiene le cifre significative. Se il punto decimale manca, è implicito che sia a destra
	E	lettera fissa che significa Esponente
	±	è il segno dell'esponente
	ee	è l'esponente espresso con una o due cifre. Esso indica l'ampiezza di tutto il numero; può essere interpretato come numero di spostamenti del punto decimale: verso sinistra (esponente negativo) per numeri piccoli, verso destra (esponente positivo) per numeri grandi

Ecco alcuni esempi:

Forma standard	**Notazione scientifica**
1000000	1 E + 06
.0000001	1 E – 07
–.00000123456789	–1.23456789 E – 06
–123456789	–1.23456789 E + 08

Come potete vedere la notazione scientifica è molto utile per rappresentare numeri molto grandi oppure molto piccoli. I valori massimo e minimo per i numeri reali, che più sopra abbiamo scritto con una infinità di zeri, possono invece essere indicati con 1 E + 38 e – 1 E + 38. Il numero più vicino allo zero può essere invece scritto come 3 E – 38.

Arrotondamento

precedentemente abbiamo detto che un numero reale può avere sino a nove cifre di precisione. Questo significa che un numero maggiore di 1 o minore di – 1 può avere solo le nove cifre più a sinistra diverse da zero. L'Apple II arrotonda le altre cifre in eccesso. Ecco alcuni esempi dove si vede anche che i numeri molto grandi sono scritti con notazione scientifica:

```
]PRINT 1234567891
1.23456789E+09

]?-123456789123456789
-1.23456789E+17

]?-150000475.75
-150000476

]?90000000.7558
90000000.8
```

I numeri frazionari (compresi tra 1 e — 1) vanno soggetti alle stesse limitazioni. In questo caso le nove cifre significative iniziano dalla prima non nulla dopo il punto decimale. Ecco alcuni esempi:

```
]PRINT .1234567891
.123456789

]?-123456789123456789
-1.23456789E+17

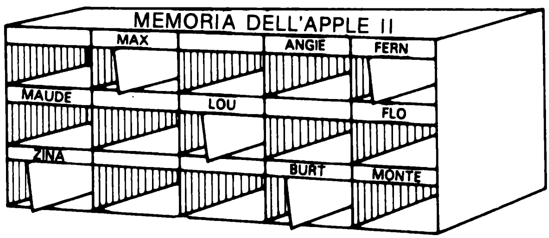
]?-123456789 123456789
-1.23456789E+17

]?.000000000900000007558
9.00000008E-10
```

VARIABILI

Sino a questo punto abbiamo considerato solamente dati numerici costanti, ma è molto comodo dar loro invece dei nomi. Questi nomi sono appunto le *variabili.*

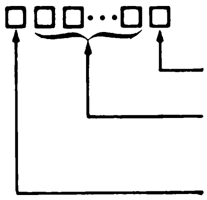
Se avete studiato l'algebra non avrete alcuna difficoltà a comprendere che cosa sono le variabili. Diversamente pensate alle variabili come dei nomi posti sulle cassette della posta. Qualunque cosa sia inserita nella cassetta diviene il valore da associarsi a quella variabile in quel momento. Nella terminologia dei calcolatori si dice che un valore è memorizzato in una variabile.



Una variabile non deve avere sempre lo stesso valore, ma può benissimo cambiarlo durante l'esecuzione del programma. Questo è il motivo della grande utilità delle variabili. Vedremo in seguito quali sono le istruzioni BASIC per cambiare il valore alle variabili.

Nomi delle variabili in Integer BASIC

In Integer BASIC i nomi delle variabili possono avere da 1 sino a 100 caratteri. Le regole sintattiche per formare questi nomi sono le seguenti:



L'ultimo carattere deve essere $ per indicare le stringhe

Dal secondo carattere in avanti (se presenti) vi possono essere lettere o cifre

Il primo carattere deve essere una lettera.

Così l'ultimo carattere dice all'interprete dell'Integer BASIC quale tipo di dato la variabile rappresenta (stringa o numero).

Le *variabili di stringa* possono riferirsi a stringhe con lunghezza da 0 a 255 caratteri. Gli spazi vuoti contano nel calcolo della lunghezza della stringa. Prima di usare una variabile di stringa dovete dichiarare la massima lunghezza che può avere con una istruzione DIM. (Questa istruzione verrà discussa in seguito). Se non fate questa dichiarazione ottenete l'errore *** STR OVFL ERR. Ecco alcuni esempi di nomi di stringa validi e illegali.

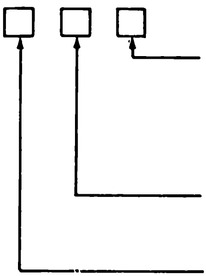
Valido	**Illegale**
A$	$
CLIENTE$	9$
X15$	CARLO.ROSSI$
PARTE58$	

Le *variabili numeriche* in Integer BASIC possono avere valori compresi tra − 32767 e 32767. Se uscite da questi limiti otterrete l'errore *** > 32767 ERR. Ecco alcuni esempi validi e illegali di nomi di variabili numeriche in Integer BASIC:

Valido	**Illegale**
A	3X4
CODICECLIENTE	L'ACQUA
X0	$TOTALE

Nomi delle variabili in Applesoft

In Applesoft un nome di variabile può avere sino a tre caratteri. Ecco le regole di formato:



L'ultimo carattere definisce il tipo di variabile:
$ per stringa,
% per variabile intera,
negli altri casi è una variabile reale.

Se presente, il secondo carattere può essere una lettera o una cifra.

Il primo carattere deve essere una lettera.

Anche in questo caso l'ultimo carattere individua il tipo di variabile.

Una *variabile di stringa* in Applesoft può memorizzare una stringa con lunghezza da 0 a 255 caratteri; non è necessario dichiarare all'inizio la massima lunghezza della stringa. Ecco alcuni esempi:

Valido	**Illegale**
A$	4$
MN$	M!$
F6$	77$

Le *variabili intere* possono rappresentare numeri interi compresi tra – 32767 e 32767. Se superate questi limiti otterrete l'errore ?ILLEGAL QUANTITY ERROR. Se tentate di porre un numero reale in un variabile intera, l'Applesoft converte il numero reale in numero intero. Le regole per questa conversione saranno viste tra breve. Ecco alcuni esempi di variabili intere in Applesoft:

Valido	**Illegale**
A%	A$
B%	37%
A5%	3D%
X6%	

Le *variabili reali* possono riferirsi a numeri compresi tra -10^{38} e $+10^{38}$. (In realtà potete arrivare a numeri di ampiezza sino a 1.7×10^{38}). Se superate questi limiti otterrete l'errore ?OVERFLOW ERROR.

Quando il valore di una variabile reale è più vicino a zero di $\pm 2.9388 \times 10^{-39}$ l'Applesoft lo converte a zero.

Ricordatevi poi che una variabile reale può avere un valore intero perchè gli interi sono numeri reali con parte frazionaria nulla. Ecco alcuni esempi:

Valido	**Illegale**
A	7B
A1	45
AA	A #
B	
Z5	

Nomi lunghi di variabili in Applesoft

I nomi delle variabili potrebbero essere più lunghi di due caratteri (più il segno $ o %), ma solamente questi primi due vengono riconosciuti dall'Applesoft. Per esempio PREZZO1 e PREZZO2 sono equivalenti a PR. PREZZO1 e PREZZO1% sono invece diversi per la presenza del suffisso %.

L'Applesoft permette che i nomi delle variabili siano lunghi fino a 238 caratteri. Ecco alcuni esempi di nomi con lunghezza estesa:

Valido	**Illegale**
CONTATORE%	VOCE # %
SALDO	2POSTO
NOME$	CLIENTE.INDIRIZZO$

Quando usate nomi lunghi per le variabili ricordatevi questi suggerimenti:

1. Solo i primi due caratteri (più il suffisso % o $) sono significativi. Non estendete i nomi pensando di diversificarli; LOOP1% e LOOP2% sono eguali a LO%!

2. La presenza di nomi estesi permette di leggere con più facilità il programma, ma contemporaneamente questi nomi occupano più memoria del calcolatore. Sta a voi stabilire se avete memoria sufficiente per usare nomi estesi o preferite risparmiarla per poter scrivere programmi più lunghi.

Parole riservate

Tutte le parole che definiscono qualche operazione del linguaggio BASIC sono dette *parole riservate.* Nell'Appendice F trovate l'elenco di tutte le parole riservate del BASIC e dell'Applesoft. Alcune di queste parole vi sono già note, altre le incontrerete in seguito.

Quando l'Apple II esegue una istruzione cerca per prima cosa le parole riservate. L'unica eccezione riguarda i testi tra virgolette che vengono presi così come sono e non interpretati. Attenzione quindi che se una parola riservata è stata inserita nel nome di una variabile, il calcolatore può essere tratto in inganno e commettere gravi errori. È molto importante quindi che *non poniate mai inavvertitamente le parole riservate nei nomi delle variabili*; ciò è facile che succeda specialmente nel caso delle parole riservate molto corte.

VARIABILI CON INDICI

Le *variabili con indici* sono concettualmente molto semplici. Quando avete più variabili che rappresentano grandezze con qualche cosa in comune, ma ciò non è essenziale, invece di dare loro nomi tutti diversi, potete dare lo stesso nome e distinguerle mediante un indice.

Consideriamo per esempio una tabella che contenga 200 numeri. Potreste semplicemente pensare di dare a ciascuno dei 200 elementi della tabella un nome diverso, ma è molto più efficace dare a tutti lo stesso nome e distinguerli proprio per la posizione che hanno nella tabella.

Consideriamo un altro esempio. Un albergo ha 10 stanze e deve registrare ogni giorno i nomi degli ospiti. Questo potrebbe essere fatto così:

JONES	SMITH	DOE		LITTKE	ALTON	DAVIS	HANSON	SHORTEN	
R1$	R2$	R3$	R4$	R5$	R6$	R7$	R8$	R9$	R10$

Oppure potete pensare di usare una variabile con indici (un indice sòlo in questo caso) e impostare le cose in quest'altro modo:

JONES	SMITH	DOE		LITTKE	ALTON	DAVIS	HANSON	SHORTEN	
R$(1)	R$(2)	R$(3)	R$(4)	R$(5)	R$(6)	R$(7)	R$(8)	R$(9)	R$(10)

In questo caso R$ è il nome di una variabile con indici. Essa ha dieci *elementi* e ogni elemento è il nome di un ospite. Un indice posto tra parentesi individua ogni elemento della variabile. Ogni nominativo è quindi individuato dal nome della variabile e dall'indice. Per esempio SMITH è il valore di R$ (2).

In questo esempio si è mostrata una variabile di stringhe con indici. Come è facile capire in Integer BASIC le variabili con indici possono contenere solo numeri interi.

Nell'Applesoft queste variabili possono invece trattare sia stringhe, che numeri reali, che numeri interi. È importante precisare subito però che i tre tipi di dati *non* possono essere mescolati in una stessa variabile. (Salvo che un numero intero sia rappresentato come reale!). Il motivo di questo è che ogni tipo di variabile con indici usa una diversa quantità di memoria per ogni suo elemento. Per i dettagli vedere l'Appendice G.

Dimensioni delle variabili con indici

In Integer BASIC dovete sempre specificare il numero degli elementi di una variabile con indici, prima di poterla usare. Questa dichiarazione viene fatta con una istruzione DIM *(dimensione)* che presenteremo più oltre in questo capitolo.

Con l'Applesoft potete invece usare variabili con indici che abbiano fino a 10 elementi senza doverle prima dimensionare.

L'Applesoft permette anche di avere variabili con più di un indice o in altre parole con più dimensioni. Una variabile con un solo indice equivale ad una tabella con una sola riga di numeri. L'indice individua la posizione di ogni numero. Una variabile con due indici equivale invece ad una tabella con più righe di numeri. Un indice individua le righe, mentre l'altro individua le colonne.

È possibile anche immaginare una variabile con tre indici che corrisponde quindi ad un cubo. Se però tentiamo di immaginare una variabile con quattro o più indici ci accorgiamo che per la nostra mente è impossibile "vederla", mentre non presenta assolutamente alcuna difficoltà trattarla matematicamente.

Per fare un esempio di variabile con due indici estendiamo l'esempio precedente dell'albergo. Supponiamo che questo albergo abbia otto piani, abbiamo quattro possibilità per impostare l'elenco degli ospiti. Primo, dare ad ogni camera un nome di va-

riabile separato. Secondo, definire una variabile con un solo indice a 80 elementi. Terzo, definire otto diverse variabili con un indice ciascuna per ogni piano. Quarto, definire una variabile con due indici: uno per il piano e l'altro per la camera. Questa scelta può essere così rappresentata:

H$(8,1)	H$(8,2)	H$(8,3)	H$(8,4)	H$(8,5)	H$(8,6)	H$(8,7)	H$(8,8)	H$(8,9)	H$(8,10)
H$(7,1)	H$(7,2)	H$(7,3)	H$(7,4)	H$(7,5)	H$(7,6)	H$(7,7)	H$(7,8)	H$(7,9)	H$(7,10)
H$(6,1)	H$(6,2)	H$(6,3)	H$(6,4)	H$(6,5)	H$(6,6)	H$(6,7)	H$(6,8)	H$(6,9)	H$(6,10)
H$(5,1)	H$(5,2)	H$(5,3)	H$(5,4)	H$(5,5)	H$(5,6)	H$(5,7)	H$(5,8)	H$(5,9)	H$(5,10)
H$(4,1)	H$(4,2)	H$(4,3)	H$(4,4)	H$(4,5)	H$(4,6)	H$(4,7)	H$(4,8)	H$(4,9)	H$(4,10)
H$(3,1)	H$(3,2)	H$(3,3)	H$(3,4)	H$(3,5)	H$(3,6)	H$(3,7)	H$(3,8)	H$(3,9)	H$(3,10)
H$(2,1)	H$(2,2)	H$(2,3)	H$(2,4)	H$(2,5)	H$(2,6)	H$(2,7)	H$(2,8)	H$(2,9)	H$(2,10)
H$(1,1)	H$(1,2)	H$(1,3)	H$(1,4)	H$(1,5)	H$(1,6)	H$(1,7)	H$(1,8)	H$(1,9)	H$(1,10)

Così R$ (3,2) indica il nome dell'ospite della seconda camera del terzo piano.

L'Applesoft permette di avere variabili con 88 indici (cioè 88 dimensioni). Non vi è alcun specifico limite al numero di elementi che vi possono essere in ogni dimensione; l'unico limite è dovuto alla memoria disponibile in quanto ogni elemento della variabile occupa ovviamente alcune posizioni di memoria.

ESPRESSIONI

In questo paragrafo analizziamo come comporre variabili e costanti in una *espressione*. Un esempio di espressione lo abbiamo già incontrato quando abbiamo scritto:

```
PRINT 4+6
10
```

per dire al calcolatore di fare la somma di 4 e 6 e poi visualizzare il risultato. Quest'altra istruzione:

```
PRINT A+B
0
```

dice invece al calcolatore di sommare il contenuto delle variabili A e B e poi visualizzare il risultato.

Il segno (+) indica una somma. Nella terminologia dei calcolatori esso rappresenta un *operatore* e in questo caso un *operatore aritmetico.*

Gli operatori aritmetici sono molto semplici da comprendere, ma esistono altri operatori che richiedono una maggiore attenzione. Essi sono gli *operatori Booleani,* gli *operatori relazionali* e gli *operatori tra stringhe.*

Ogni tipo di operatore definisce un corrispondente tipo di espressione. Abbiamo così espressioni aritmetiche, espressioni tra stringhe, espressioni relazionali e espressioni Booleane.

Criteri di precedenza degli operatori nelle espressioni

In una espressione possono comparire più operatori. Per esempio in questa espressione:

```
PRINT A+B/10
0
```

compaiono sia l'operatore somma che quello divisione. Si pone allora il problema di stabilire con quale ordine devono essere eseguiti i vari operatori. Nei paragrafi che seguono daremo le regole di *precedenza* degli operatori iniziando dal concatenamento delle stringhe, trattando poi le espressioni con interi, reali, relazionali, Booleani e da ultimo le espressioni di tipo misto.

Uso delle parentesi

Prima di descrivere i criteri di precedenza degli operatori, vediamo che mediante l'uso di parentesi è possibile imporre l'ordine che noi riteniamo più opportuno.

Ogni operazione posta fra parentesi viene eseguita per prima. Se vi sono più coppie di parentesi vengono eseguite da sinistra verso destra.

Se più coppie di parentesi sono poste l'una dentro l'altra, viene eseguita prima la più interna. In questo caso si dice che le parentesi sono *nidificate (nested).* Le parentesi possono essere nidificate ad ogni livello per rendere esplicito l'ordine con cui devono essere eseguite le operazioni in una espressione.

Ecco alcuni esempi di operazioni aritmetiche poste tra parentesi:

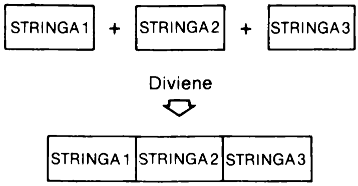
```
PRINT (2+10)*3
36

>
PRINT ((2+10)*3+31)*10
670

>
PRINT -(2^(3+8/4))
-32
```

Concatenamento di stringhe

Potete unire più stringhe una di seguito all'altra per formare una stringa più lunga. Questa operazione viene detta *concatenamento.* Ecco un esempio esplicativo:



Mediante concatenamento potete creare stringhe lunghe sino a 255 caratteri.

In Integer BASIC non esiste questa funzione concatenamento di stringhe, anche se è possibile unire più stringhe con una tecnica che spiegheremo alla fine di questo capitolo.

Con l'Applesoft è invece possibile usare l'operatore (+) come indicato in questi esempi:

"SCA" + "DUTO"	diviene	"SCADUTO"
"RAPPORTO" + " " + "MENSILE"	diviene	"RAPPORTO MENSILE"
"MESE" + R$	diviene	la parola MESE seguita dal contenuto di R$
A$ + B$ + C$(1)	diviene	il valore di A$ unito a quello di B$ e a quello di C$(1)

Espressioni intere

Le espressioni intere sono espressioni aritmetiche che comprendono unicamente variabili e costanti intere. In seguito definiremo anche espressioni di tipo misto che possono trattare sia valori reali che interi.

Gli operatori per le espressioni intere sono: somma (+), sottrazione (–), moltiplicazione (*), divisione (/) e potenza (^). È possibile anche usare un segno meno (–) per indicare valori negativi. Le operazioni aritmetiche sono eseguite con questo ordine: prima il (–) per indicare i valori negativi, poi le potenze, quindi le moltiplicazioni e divisioni e infine le addizioni e le sottrazioni. Operazioni di eguale importanza sono eseguite da sinistra verso destra.

Ecco alcuni esempi di espressioni intere in Integer BASIC:

100 – 30 * 2	risultato	40
–9^2	risultato	81
A/B * C	risultato	il valore di A diviso per il valore di B e la parte intera del quoziente moltiplicata per C
D + X * 3	risultato	tre volte il valore di X più il valore di D
5/2 * 2	risultato	4; la parte intera di 5/2 moltiplicata per 2

Queste invece sono espressioni intere in Applesoft:

−120/2 + 100	risultato	40
2^3 * 2	risultato	16
N1% * N2% * N3%	risultato	il valore di N1% volte il valore di N2% volte il valore di N3%
AA%/AB%/AC%	risultato	il valore di AA% diviso per il valore di AB% e il quoziente diviso per AC%
5/2 * 2	risultato	5; (il quoziente 5/2 non viene convertito in intero)

In Integer BASIC esiste anche l'operatore modulo MOD che fornisce il resto della divisione tra due numeri tra loro non divisibili. L'operatore MOD ha la stessa precedenza delle moltiplicazioni e divisioni. Ecco alcuni esempi:

4 MOD 3	risultato 1
3 * 5 MOD 4	risultato 3
3 MOD 4	risultato 3

Espressioni reali

In Applesoft le espressioni aritmetiche possono essere anche reali. Gli operatori sono gli stessi di quelle intere (+, −, *. /, ^e segno −) e seguono gli stessi criteri di precedenza: prima il segno (−), poi le potenze, poi le moltiplicazioni e le divisioni e infine le addizioni e sottrazioni. Ecco alcuni esempi di espressioni reali:

87.5 − 4.25 * 2	risultato	79
1.5^(3/2/2)	risultato	1.35540301
AL * (PL − 3.1 * CB)	risultato	il valore di AL volte il risultato della sottrazione del prodotto, di 3.1 per il valore di CB, dal valore di PL
7.5 * 2/5	risultato	3

Espressioni relazionali

Gli operatori relazionali permettono di fare un confronto tra due valori. I valori che si possono confrontare sono sia costanti, che variabili, che qualunque tipo di espressione, salvo qualche eccezione in Integer BASIC. Gli operatori relazionali sono: maggiore di, minore di, eguale a, diverso da, maggiore o eguale a e infine minore o eguale a. Se uno dei due valori confrontati è una stringa, anche l'altro deve essere una stringa.

Se la relazione è vera, l'espressione relazionale assume il valore 1. Se la relazione è falsa l'espressione assume il valore 0.

Gli operatori relazionali sono gli stessi sia in Integer BASIC che in Applesoft, salvo alcune eccezioni come indicato nella Tabella 3.1.

Tabella 3.1 – Operatori relazionali

Operatori in Integer BASIC	**Funzione**	**Operatori in Applesoft**
<	Minore di*	<
>	Maggiore di*	>
=	Eguale a	=
#	Diverso da	< > o > <
>=	Maggiore o eguale a*	> = o = >
< =	Minore o eguale a*	< = o = <
* Non consentito con le stringhe in Integer BASIC.		

Tutti gli operatori relazionali hanno la stessa precedenza e sono calcolati da sinistra verso destra.

Diamo qui di seguito alcuni esempi di espressioni relazionali:

1 = 5 − 4	vero (1)	
14 > 66	falso (0)	
15 > = 15	vero (1)	
"AA" = "AA"	vero (1)	
"CARLO" > "ALDO"	vero (1)	
(A = B) = (A$ > B$).		dipende dal valore delle variabili; se il valore di A è uguale al valore di B e il valore di A$ è maggiore del valore di B$, allora il risultato è vero (1)

È importante ora capire che il valore 0 o 1, che il BASIC assegna alle espressioni relazionali, può essere usato nelle espressioni aritmetiche. Forse vi chiederete che significato può avere una espressione come (1 = 1) * 4? Fuori dal BASIC probabilmente non significa nulla, ma in BASIC (1 = 1) è vero e quindi vale 1 per cui l'espressione di prima è come 1 * 4 cioè 4. Ecco altri esempi esplicativi:

25 + (14 > 66)	equivale a: 25 + 0
(A + (1 = 5 − 4)) * (15 >= 15)	equivale a: (A + 1) * (1).

Confronto tra stringhe

Forse vi sarete chiesti quali regole il calcolatore Apple II usa per confrontare due stringhe. Possiamo fare due considerazioni. La prima riguarda la lunghezza delle stringhe, compresi gli spazi, e dice che se due stringhe hanno diversa lunghezza non possono essere eguali. Se la prima parte della stringa più lunga è uguale a quella più corta, allora la stringa più lunga è maggiore della più corta (questo tipo di confronto riguarda però solamente l'Applesoft).

La seconda considerazione riguarda il fatto se le due stringhe contengono gli stessi caratteri nello stesso ordine.

Le stringhe sono confrontate un carattere alla volta iniziando da quelli più a sinistra. Se i primi due sono eguali si passa a confrontare i secondi due e così via sino a che una delle due stringhe termina oppure due caratteri sono diversi.

L'Applesoft confronta la posizione relativa dei caratteri uno alla volta. Per convenzione, durante i confronti, le lettere hanno questo ordine: $A<B<C<D$ ecc. I numeri che compaiono nelle stringhe hanno l'ordine $0<1<2<3<4$ ecc.. I caratteri speciali come +, –, $ ecc. hanno anche loro un ordine come indicato nell'Appendice I.

Tabella 3.2 – Tavola della verità Booleana

Il risultato di AND è 1 solo se ambedue i valori sono 1

1 AND 1 = 1	1 AND 0 = 0
0 AND 1 = 0	0 AND 0 = 0

Il risultato di OR è 1 se uno dei due valori è 1

1 OR 1 = 1	1 OR 0 = 1
0 OR 1 = 1	0 OR 0 = 0

Il risultato di NOT è il complemento del valore

NOT 1 = 0
NOT 0 = 1

Espressioni Booleane

Mediante gli operatori Booleani un programma può prendere delle decisioni logiche. Spesso gli operatori Booleani sono anche chiamati *operatori logici.* Generalmente esistono quattro operatori standard: AND, OR, OR Esclusivo e NOT. Il BASIC dell'Apple II usa solo tre di questi operatori: AND, OR e NOT.

Se non conoscete già gli operatori Booleani cerchiamo di chiarirvi il loro significato con un esempio. Supponiamo che dobbiate comperare dei biscotti per due bambini. L'operatore AND dice che li comprerete se ambedue i bambini vogliono i biscotti. L'operatore OR dice che li comprerete se uno almeno dei due bambini li vuole (ma

anche tutti e due i bambini li possono volere). L'operatore NOT genera una negazione. Se un bambino li vuole e l'altro no, allora si dice che la decisione di un bambino è "NOT" quella dell'altro.

Gli operatori Booleani, analogamente a quelli relazionali, giungono a risultati che possono essere veri (1) o falsi (0).

La Tabella 3.2, che viene anche chiamata *tavola della verità,* riporta i valori che assumono gli operatori Booleani.

Se più operatori Booleani sono presenti in una espressione, essi hanno eguale precedenza e vengono calcolati da sinistra verso destra. Ecco alcuni esempi di espressioni Booleane:

NOT ((3+4) > = 6)	risultato	0 (falso)
("AA" = "AB") OR ((8 * 2) = 4^2)	risultato	1 (vero)
NOT ("APPLE" = "ARANCIO") AND (A$ = B$)	risultato	1 (vero) se A$ e B$ sono eguali; 0 (falso) se sono diversi

Espressioni di tipo misto

Molto spesso le espressioni non contengono valori di un solo tipo. Questo può facilmente accadere nell'Applesoft in cui si possono avere sia numeri interi che reali. Abbiamo già incontrato espressioni di tipo misto parlando di espressioni Booleane e relazionali. Sappiate quindi che in una espressione potete usare valori di qualunque tipo eccetto le stringhe. Le stringhe possono infatti essere presenti solo in espressioni di stringa o relazionali. Ecco alcuni esempi di espressioni di tipo misto:

Valido	**Illegale**
3.1416 * (R^2)	1600 + "VIA VENETO"
A% > = B/3	ST$ < A%
43 AND 137	A$ AND B$
1 OR 4E + 10	NOT(A$) = B$
(A$ = B$) AND − 6.25	NOT(A = B) OR C$

Il calcolatore Apple II, quando incontra una espressione di tipo misto, individua dapprima le precedenze degli operatori. In Tabella 3.3 sono riportati tutti gli operatori in ordine di precedenza di esecuzione. Potete vedere che ogni cosa contenuta tra parentesi viene eseguita per prima. Se vi sono più coppie di parentesi "nidificate" il calcolatore esegue prima il contenuto delle parentesi più interne, poi le successive sempre più verso l'esterno. Poi seguono gli operatori aritmetici, poi quelli relazionali ed infine quelli Booleani.

Tabella 3.3 — Operatori

	Precedenza	Operatori in Integer BASIC	Operatori in Applesoft BASIC	Significato
	Massima 9	( )	( )	Le parentesi impongono l'ordine di elaborazione
Operatori Aritmetici	8	^	^	Potenza
	7	–	–	Meno unario
	6	*	*	Prodotto
	6	/	/	Divisione
	6	MOD	non disponibile	Resto della divisione**
	5	+	+	Somma
	5	–	–	Sottrazione
Operatori Relazionali	4	=	=	Eguale
	4	#	< >o> <	Diverso da
	4	<	<	Minore di
	4	>	>	Maggiore di
	4	< =	< =o= <	Minore o uguale a
	4	> =	> =o= >	Maggiore o uguale a
Operatori Booleani	3	NOT	NOT	Complemento logico
	2	AND	AND	AND logico
	1 Minima	OR	OR	OR logico

** Solo in Integer BASIC

Come abbiamo già visto precedentemente, le espressioni relazionali "ritornano" i valori 0 o 1 a seconda che la relazione sia falsa o vera. Di conseguenza una espressione relazionale può far parte di una espressione intera o reale.

Anche nelle espressioni Booleane potete mescolare i tipi. In queste espressioni qualunque cosa viene dapprima convertita in 0 o 1 prima di eseguire gli operatori logici. I valori numerici sono convertiti secondo queste regole: se il valore numerico è 0 esso rimane 0; ogni altro valore diverso da 0 viene convertito in 1.

Il BASIC non può convertire automaticamente le stringhe in valori numerici. Per questo motivo le stringhe sono illegati nelle espressioni intere, reali e Booleane eccetto che nelle espressioni relazionali.

In Applesoft sia i valori reali che quelli interi possono essere presenti in una espressione di tipo reale, relazionale o Booleano.

In una espressione reale, se sono presenti dei valori interi, essi sono convertiti temporaneamente in forma reale per potere eseguire il calcolo dell'espressione. Il risultato finale di questa espressione può essere sia un valore intero che uno reale e ciò dipende dal contesto in cui è presente l'espressione. L'Applesoft converte il risultato automaticamente nella forma più opportuna.

I valori reali sono convertiti in intero elidendo la parte frazionaria e prendendo il numero intero immediatamente inferiore. Questa operazione si chiama *troncamento* e qui di seguito diamo alcuni esempi:

1.1	diviene	1
1.9	diviene	1
–1.1	diviene	–2
–1.9	diviene	–2

ISTRUZIONI BASIC

Iniziamo ora la descrizione delle istruzioni BASIC. Spesso si usano alternativamente le parole *istruzioni* e *comandi* per indicare i singoli passi successivi di un programma. Sarebbe più esatto però usare la parola comando per quelle istruzioni che sono date in modo immediato e riservare la parola istruzione nel caso del modo differito.

Ogni istruzione descrive un particolare passo del programma. In questo capitolo non vogliamo entrare nei dettagli delle singole istruzioni, ma darvi una visione d'insieme di come si usano le istruzioni cioè di come si fa a programmare. Nel capitolo 8 potete trovare invece la descrizione esatta e completa di ognuna di esse.

Se desiderate poi approfondire particolarmente le tecniche di programmazione potete consultare i testi che abbiamo elencato nell'Appendice K.

COMMENTI

Esiste in BASIC una istruzione che non viene mai eseguita: la REM (dall'inglese REMARK). La REM non rappresenta niente di operativo e viene completamente trascurata dall'interprete BASIC. Essa ha unicamente lo scopo di permettere di inserire delle frasi di commento direttamente nel programma. L'importanza di queste frasi non deve essere trascurata perchè esse permettono di rendere facilmente leggibile un programma.

Se infatti scrivete un programma lungo cinque o dieci righe non avrete mai alcuna difficoltà a ricordare che cosa esso faccia anche dopo sei mesi che lo avete scritto. Ma se scrivete un programma lungo cento o duecento righe, vi sarà molto facile dimenticare qualche suo punto molto importante. Se poi scrivete molti programmi, allora potete addirittura confonderne uno con un altro. L'unico modo per ovviare a questi inconvenienti è quello di documentare il programma cioè di inserirvi dei commenti.

I programmatori professionisti inseriscono moltissimi commenti nei loro programmi. In questo capitolo anche noi ne inseriremo spesso per abituarvi ad usarli.

Le istruzioni REM devono avere il loro numero di linea come qualunque altra istruzione. Questi numeri possono a loro volta essere usati come tutti gli altri numeri di linea.

ISTRUZIONI DI ASSEGNAZIONE

Le istruzioni di assegnazione permettono di attribuire dei valori alle variabili. In un programma BASIC queste istruzioni sono forse quelle più frequenti.

Ecco un esempio di istruzione di assegnazione:

```
90 REM INIZIALIZZAZIONE VARIABILE X
100 LET X=3
```

L'istruzione 100 assegna il valore 3 alla variabile X. La stessa istruzione può essere scritta così:

```
100 X=3
```

La parola LET è opzionale e può non essere usata. Ecco ora un esempio di assegnazione di stringa:

```
215 A$="ALSO RAN"
```

Alla stringa A$ viene attribuito il valore "ALSO RAN". Mostriamo ora come fare per assegnare valori agli elementi di una variabile con indice R$ ( ):

```
200 REM R$() LISTA OSPITI ALBERGO
210 R$(1)="JONES"
220 R$(2)="SMITH"
230 R$(3)="DOE"
```

Ricordatevi però che è possibile porre su una stessa linea più di una istruzione per risparmiare memoria; le ultime istruzioni di assegnazione possono allora essere riscritte così:

```
200 REM R$() LISTA OSPITI ALBERGO
210 R$(1)="JONES":R$(2)="SMITH":
R$(3)="DOE"
```

Attenzione! Mettete i due punti (:) per separare le singole istruzioni su una stessa linea.

Una istruzione di assegnazione può contenere qualunque operatore aritmetico o logico descritto precedentemente. Ecco un esempio:

```
90 REM ESEMPIO DI INIZIALIZZAZIONE
100 V=33+7/9
```

Il valore 4.17647059 viene assegnato alla variabile V; la stessa istruzione può essere scritta in forma più elementare così:

```
90 REM X E Y SONO INIZIALIZZATE
95 REM SEPARATAMENTE
100 X=7
110 Y=9
120 V=33+X/Y
```

oppure su una stessa linea:

```
100 X=7:Y=9:V=33+X/Y
```

Diamo ora alcuni esempi di istruzioni di assegnazione che contengono operatori Booleani:

```
90 REM ESEMPI GIA' DESCRITTI
95 REM IN PRECEDENZA
100 A=NOT((3+4)>=6)
110 B=("AA"="AB") OR ((8*2)=(4^2))
```

Questo è invece un esempio di assegnazione con concatenamento di stringhe:

```
90 REM A R$(6) VIENE ASSEGNATO
95 REM IL VALORE MR. ALTON
100 MR$="MR. "
110 MS$="MS. "
120 N$="ALTON"
200 R$(6)=MR$+N$
```

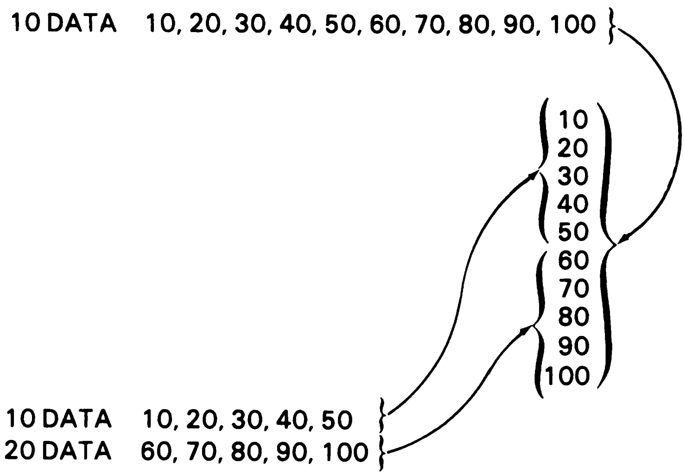
Istruzioni DATA e READ

In Applesoft è possibile assegnare il valore a più variabili direttamente con la coppia di istruzioni DATA e READ, invece di fare delle singole assegnazioni per ogni variabile. Osservate infatti questo esempio:

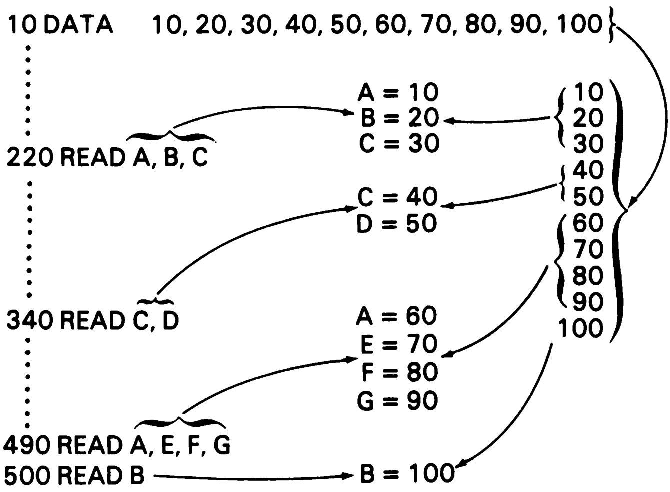
```
5 REM INIZIALIZZAZIONE DI TUTTE LE
7 REM VARIABILI DEL PROGRAMMA
10 DATA 10, 20, -4, 300
20 READ A,B,C,D
```

L'istruzione alla linea 10 specifica quattro dati numerici. Questi quattro valori sono poi assegnati alle quattro variabili della linea 20. Dopo l'esecuzione di questa linea si ottiene: A = 10, B = 20, C = –4 e D = 300.

In un programma vi possono essere più istruzioni DATA, ma ciò equivale alla presenza di una sola. Per farvi comprendere questo immaginate una istruzione DATA con 10 valori. Invece di scrivere questa sola istruzione, possiamo scriverne due che contengano i primi cinque e i secondi cinque valori. Qui di seguito cerchiamo di chiarirvi questo concetto graficamente:



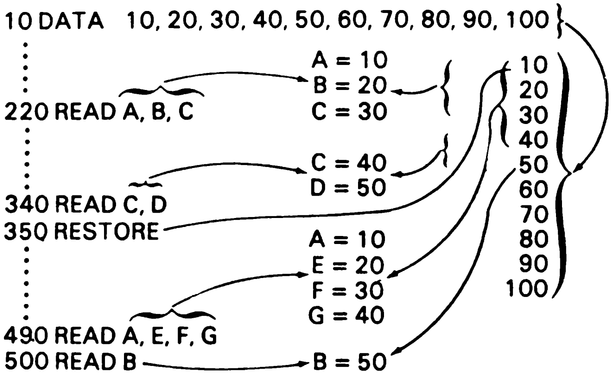
Anche le istruzioni READ possono essere più di una. La prima assegna alle sue variabili i primi valori che compaiono nella colonna di valori delle DATA. La seconda READ, come anche le successive, assegnano con ordine i valori che sono rimasti disponibili. Ciò può essere illustrato così:



L'istruzione DATA può contenere valori sia numerici che di stringa, ma quando la READ li va a prelevare deve farli corrispondere a variabili dello stesso tipo (stringhe o numeri).

Istruzione RESTORE

Per controllare l'ordine con cui vengono prelevati i valori delle DATA, possiamo immaginare esista un *puntatore* che indichi, di volta in volta, quale è il prossimo valore che può essere letto da una READ. Mediante l'istruzione RESTORE (solo in Applesoft) questo puntatore viene riportato all'inizio della colonna così che tutti i valori possono essere riletti dalle istruzioni READ successive.



Azzeramento delle variabili

È possibile sia in Integer BASIC che in Applesoft, portare tutte le variabili numeriche a zero e annullare tutte le variabili di stringa.

In Integer BASIC dovete usare CLR, ma solo in modo immediato: Ecco un esempio:

```
>X=37

>PRINT X
37

>CLR

>PRINT X
0
```

In Applesoft l'istruzione è invece CLEAR. Tale istruzione esegue implicitamente anche un comando RESTORE. Ecco un esempio:

```
]10 REM INIZIALIZZAZIONE DELLE VARIABILI
]20 X=37
]30 A$="PEPPINO"
]40 PRINT A$
]50 CLEAR
]60 PRINT X

]RUN

PEPPINO
0
```

DICHIARAZIONE DELLE VARIABILI CON INDICI E DELLE STRINGHE

Se in un programma dovete usare variabili con indici oppure variabili di stringa, dovete dichiarare la loro massima ampiezza (o dimensione) mediante una istruzione DIM. L'istruzione DIM deve essere posta all'inizio del programma e può dimensionare tutte le variabili del vostro programma purchè entrino tutte su una stessa linea.

In Integer BASIC le variabili o le stringhe dimensionate possono avere una sola dimensione. Cioè le variabili numeriche possono avere un solo indice e le stringhe possono essere solo una singola sequenza di caratteri e non una struttura con più stringhe come è invece possibile in Applesoft. In questo esempio vengono dimensionate due stringhe di 5 e 25 caratteri e una variabile con indice numerica con 13 elementi (indice da 0 a 12):

```
10 DIM S1$(5),S2$(25),NB(12)
```

Il numero che segue una stringa nella DIM rappresenta il massimo numero di caratteri che la stringa può avere durante il programma. Il numero che segue una variabile numerica rappresenta invece il valore massimo che può assumere il suo indice.

Nel caso dell'Applesoft invece, appena esso incontra una variabile con indici controlla se questa variabile è già stata dimensionata. Se ciò non è stato fatto, l'Applesoft dimensiona automaticamente tale variabile riservando 11 posizioni (indice da 0 a 10) per ogni indice presente. Nell'esempio che segue vengono dimensionate due variabili con indice; una di stringhe e una numerica intera:

```
]15 DIM R$(10),R%(20)
```

La variabile con due indici che rappresentava gli ospiti di un albergo potrebbe essere dimensionata così:

```
]15 DIM H$(8,10)
```

Ma ciò non è necessario in quanto gli indici non superano il valore 10 per cui l'Applesoft fa il dimensionamento automatico. Dovete dimensionare con la DIM solo

quando prevedete che un indice assuma un valore superiore a 10. Ricordate ancora che gli indici possono iniziare dal valore 0 per cui gli elementi possibili di una variabile di questo tipo sono uno in più di quelli dichiarati nella DIM.

Nell'esempio precedente gli indici di H$ (8,10) sono due, ma se ne possono avere sino anche a 88.

Ridimensionamento

Potete dimensionare le variabili con indici una sola volta in un programma. Se dovete cambiare il dimensionamento di qualche variabile è necessario che rieseguiate l'intero programma.

Le variabili con indici non possono mai usare indici superiori a quelli dichiarati nel dimensionamento.

ISTRUZIONI DI SALTO

Come abbiamo già illustrato i programmi sono eseguiti secondo l'ordine crescente della numerazione delle linee. Mediante le istruzioni di salto è possibile cambiare questa sequenza di esecuzione.

Istruzione GOTO

L'istruzione GOTO è la più semplice istruzione di salto. Essa permette di specificare esplicitamente quale deve essere la successiva istruzione da eseguire. Consideriamo questo esempio:

```
 20 A = 4
 30 GOTO 100
 40
 50
 60
 70
 80
 90
100
110
 .
 .
 .
```

Alla linea 20 vi è una istruzione di assegnazione per attribuire il valore 4 ad A. L'istruzione successiva dice che il programma deve proseguire alla linea 100 e non alla 40.

Ovviamente in altri punti del programma devono essere presenti delle istruzioni che facciano saltare alla linea 40 perchè diversamente queste istruzioni intermedie non sarebbero mai eseguite.

Le istruzioni a cui saltare possono essere qualunque e anche quelle di commento

REM. In questo caso però ciò equivale a saltare a quella successiva. Se per esempio scrivete:

```
20 A = 4
30 GOTO 70
40
50
60
70 REM QUESTA LINEA CONTIENE SOLO COMMENTI
80
90
.
.
.
```

Il programma salta dalla linea 30 alla 70 che però non viene eseguita e quindi prosegue subito alla linea 80. Invece di saltare alla 70 potete allora saltare direttamente alla 80 come vi illustriamo qui di seguito:

```
20   A = 4
30   GOTO 80
40
50
60
70   REM
80
90
etc.
```

Se saltate ad un numero di linea che non esiste commettete un errore che viene segnalato con un messaggio.

Istruzione GOTO calcolato

Esiste un diverso tipo di istruzione GOTO che permette di eseguire il salto ad un certo numero di linea a seconda del valore assunto da una espressione numerica.

Consideriamo questa sequenza in Integer BASIC:

```
        10
        20
        30   A = B - 3
        40   GOTO 30*A + 50
A = 0   50
        60
        70
A = 1   80
        90
       100
A = 2  110
       120
       etc.
```

L'istruzione alla linea 40 è un GOTO *calcolato.* Il programma salterà alla linea 50 se A = 0, alla linea 80 se A = 1 e alla linea 110 se A = 2. Se il numero di linea calcolato non esiste viene dato l'errore * * * BAD BRANCH ERR. Nell'esempio il valore di A dipende da quello di B, ma ai fini dell'esempio non interessa sapere come B viene calcolato purchè esso assuma i valori 3, 4 o 5 per dar luogo ai salti indicati.

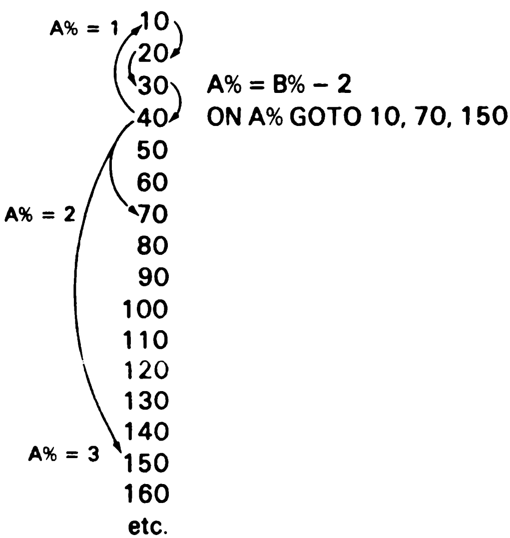
Per comprendere ancora meglio come opera il GOTO calcolato, battete questo programma di Integer BASIC:

```
>9 REM INIZIALIZZAZIONE VARIABILE B
>10 B=4
>20 PRINT B
>30 A=B-3
>40 GOTO 30*A+50
>49 REM B=3
>50 END
>79 REM B=4
>80 PRINT B
>90 B=5
>100 GOTO 20
>109 REM B=5
>110 PRINT B
>120 B=3
>130 GOTO 20
```

Ora provate ad eseguirlo battendo il comando RUN.

Avete capito con quale sequenza esso viene eseguito? Provate ora a modificarlo per fargli eseguire la sequenza 345345345...

In Applesoft il GOTO calcolato è parzialmente diverso. Osservate questo esempio:



L'istruzione alla linea 40 è il GOTO calcolato dell'Applesoft. Quando questa istruzione sarà eseguita il programma salterà alla linea 10 se A% = 1, alla linea 70 se A% = 2 e alla linea 150 se A% = 3. Se A% assume un qualunque altro valore diverso da 1, 2 o 3 il programma prosegue alla linea 50.

Il valore dell'espressione numerica determina a quale linea saltare tra quelle indicate nel GOTO calcolato dell'Applesoft. Se questo valore è 1 il programma salta al primo numero di linea, se è 2 salta al secondo numero di linea e se è 3 salta al terzo. Se l'espressione assume il valore 0 o qualunque altro valore il programma prosegue al numero di linea immediatamente successivo.

Ecco un esempio in Applesoft:

```
10 B% = 4
20  PRINT B%
30 A% = B% - 2
40  ON A% GOTO 10,70,150
70  PRINT B%
80 B% = 5
90  GOTO 30
150  PRINT B%
160 B% = 3
170  GOTO 20
```

CICLI

Le istruzioni GOTO e GOTO calcolato vi permettono di realizzare qualunque sequenza riteniate opportuna. Alcune volte però dovete rieseguire una istruzione, o un gruppo di istruzioni, parecchie volte. Se per esempio avete una variabile con indice di 100 elementi e dovete assegnare un valore a questi 100 elementi, allora potete scrivere 100 istruzioni diverse di assegnazione oppure pensare di rieseguire più volte la stessa istruzione. Questa sequenza ripetitiva viene detta *ciclo.*

Istruzioni FOR e NEXT

Con le istruzioni FOR e NEXT potete scrivere un ciclo nel modo seguente:

```
10 DIM A(99)
20 FOR I=0 TO 99 STEP 1
30 A(I)=I
40 NEXT I
```

Le istruzioni tra FOR e NEXT sono eseguite più volte. Nell'esempio sono una sola e rappresentano una assegnazione. Questa parte di programma viene detta *ciclo FOR NEXT.*

In questo esempio vengono anche visualizzati i valori assegnati alla variabile A ( ):

```
10 DIM A(99)
20 FOR I=0 TO 99 STEP 1
30 A(I)=I
35 PRINT A(I)
40 NEXT I
50 END
```

Quando battete RUN, il programma visualizza 100 numeri da 0 a 99.

Le istruzioni tra FOR e NEXT vengono eseguite tante volte come indicato dalla *variabile indice* che appare subito dopo FOR. Nell'esempio la variabile indice è la I. Alla I viene imposto di variare da 0 a 99 con *passo* (*step*) 1.

La variabile I appare anche alla linea 30. Di conseguenza quando questa istruzione di assegnazione viene eseguita per la prima volta I è eguale a 0:

```
30 A(0)=0
```

I viene incrementata di 1 ad ogni passo e così alla seconda esecuzione si ha I = 1 e quindi:

```
30 A(1)=1
```

I continua ad essere incrementata sino al valore finale che nell'esempio è 99.

Il passo può avere qualunque valore intero e non necessariamente 1. Provate a cambiare il passo alla linea 20 da 1 in 5 e rieseguite il programma. Ora il ciclo viene eseguito 20 volte. Durante l'ultimo passaggio valido la variabile I assume il valore 95; con un altro incremento di 5 diverrebbe eguale a 100 cioè oltre il valore massimo di 99 previsto nell'istruzione FOR.

Se il passo rimane eguale a 5 per eseguire allora 100 volte il ciclo bisogna spostare il valore massimo a 499. Provate a fare queste modifiche per esercizio. (Ricordatevi di cambiare anche l'istruzione DIM).

Il passo può essere anche negativo. In questo caso però il valore iniziale della variabile indice deve essere maggiore del valore finale. Per esempio se il passo è −1 e vogliamo assegnare i 100 valori di A (I) da 0 a 99, dobbiamo allora scrivere l'istruzione 20 in questo modo:

```
10 DIM A(99)
20 FOR I=99 TO 0 STEP -1
30 A(I)=I
35 PRINT A(I)
40 NEXT I
50 END
```

Se il passo è unitario, come succede normalmente, non è necessario indicarlo esplicitamente nell'istruzione FOR. In assenza di ogni indicazione il BASIC pone implicitamente il passo eguale a 1.

È possibile anche indicare i valori iniziale, finale e il passo del ciclo mediante espressioni. Dovete però cercare di evitare questa possibilità, perchè complica inutilmente il programma. Potete sempre, se necessario, calcolare con istruzioni separate i parametri da inserire nell'istruzione FOR.

Sappiate infine che gli estremi di un ciclo e il passo possono essere anche numeri reali in Applesoft.

Nell'istruzione NEXT dell'Applesoft potete non indicare la variabile del ciclo, ma è sempre preferibile farlo per rendere più leggibili i programmi.

Cicli nidificati

I cicli di programma FOR-NEXT sono molto comuni in tutti i programmi. Alcune volte è necessario anche che un ciclo contenga altri cicli. In questo caso si dice che i cicli sono *nidificati.*

Dal momento che le istruzioni contenute in un ciclo possono essere in genere di qualunque tipo, è allora possibile che tra di esse vi sia un altro ciclo FOR-NEXT.

Ecco un esempio di nidificazione primaria:

```
10 DIM A(99)
20 FOR I=0 TO 99 STEP 1
30 A(I)=I
40 REM VISUALIZZAZIONE DEI VALORI A(I)
45 REM APPENA ASSEGNATI
50 FOR J=0 TO I
60 PRINT A(J)
70 NEXT J
80 NEXT I
90 END
```

Nidificazioni molto più complesse si possono avere anche in programmi molto brevi. Vi diamo qui un esempio di nidificazione complessa senza indicare le istruzioni contenute nei cicli:

```
  50 FOR I=1 TO 10
  60 FOR X=25 TO 347 STEP 3
     ⋮
 100 FOR A=9 TO 0 STEP -1
     ⋮
 140 NEXT A
 200 FOR B=25 TO 100 STEP 5
     ⋮
 280 NEXT B
 300 NEXT X
     ⋮
 500 FOR Y=1 TO 20 STEP 2
     ⋮
 600 FOR P=10 TO 20
     ⋮
 650 NEXT P
 700 NEXT Y
     ⋮
1000 FOR Z=1 TO 10
     ⋮
1090 NEXT Z
     ⋮
1200 NEXT I
     ⋮
```

Il ciclo più esterno usa la variabile indice I; esso contiene tre cicli più interni che usano le variabili X, Y e Z. Il ciclo X contiene altri due cicli le cui variabili sono A e B. Il ciclo Y contiene il ciclo P. Il ciclo Z non contiene altro ciclo.

Le strutture nidificate di cicli sono molto semplici e facili da riconoscere. Esiste solo un errore che dovete assolutamente evitare di commettere: non terminate mai un ciclo più esterno prima di aver terminato i cicli interni! Questi esempi sono illegali:

```
 50 FOR I=1 TO 10
 60 FOR X=25 TO 347 STEP 3
    ⋮
100 NEXT I
    ⋮
200 NEXT X
```

Ogni programma deve avere un egual numero di istruzioni FOR e NEXT perchè ogni ciclo inizia con FOR e termina con NEXT.

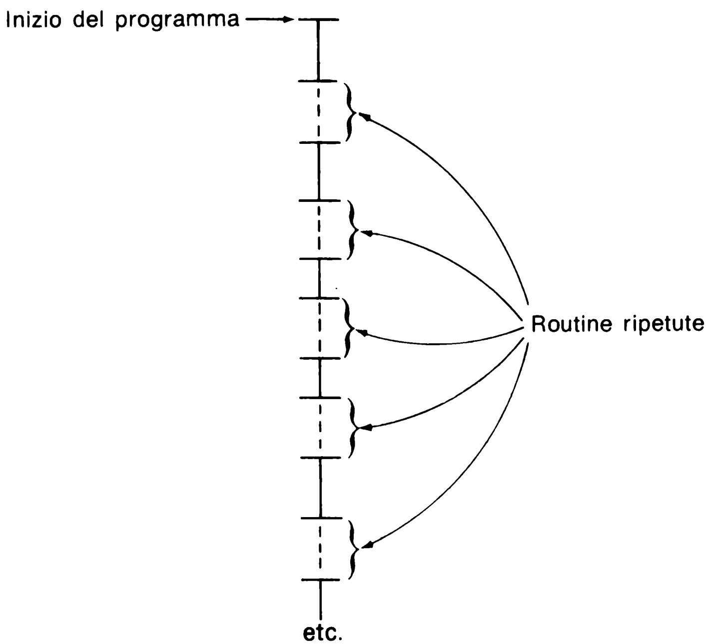
Per esempio supponete che vi sia una sola FOR e due NEXT. La prima NEXT termina correttamente l'istruzione FOR, ma la seconda NEXT non trova un corrispondente FOR e si genera quindi uno stato di errore.

In Applesoft se non ponete le variabili indice nelle NEXT i cicli vengono egualmente chiusi correttamente perchè esiste una sola possibilità logica di chiusura corretta. Osservate infatti gli esempi precedenti e vedrete che non vi è mai alcun dubbio su quale ciclo deve essere chiuso prima degli altri.

ISTRUZIONI SUBROUTINE

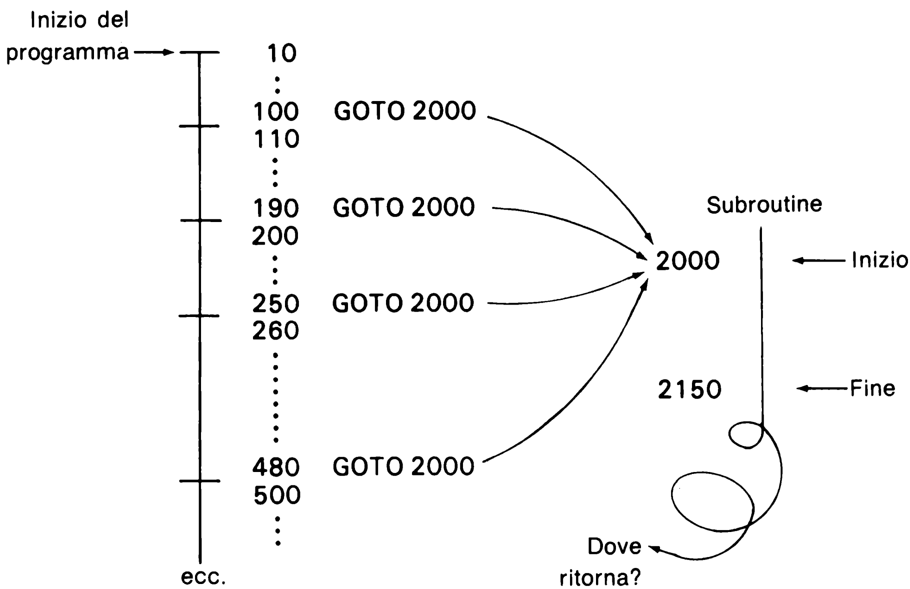
Frequentemente alcune parti di uno stesso programma devono essere utilizzate più volte. Supponiamo per esempio di avere una variabile con indice A ( ) che debba essere ripetutamente inizializzata. Possiamo allora scrivere ogni volta le tre istruzioni che costituiscono il ciclo FOR-NEXT di inizializzazione visto in precedenza. Solo in questo caso è accettabile però riscrivere le stesse istruzioni perchè sono solo tre. Diversamente, se le istruzioni fossero molte di più, sarebbe sia faticoso riscriverle sia un notevole spreco di memoria.

Il concetto che desideriamo ora illustrarvi consiste nel fatto che si possono isolare queste istruzioni e accedere ad esse con dei salti tutte le volte che è necessario utilizzarle. In questo caso il gruppo di istruzioni che abbiamo separato prende il nome di *subroutine*.



Analizzando più a fondo questo concetto di subroutine ci accorgiamo che esiste un problema. Per passare dal programma principale alla subroutine è sen'altro possibile usare una istruzione di salto dal momento che la subroutine ha un suo preciso numero di linea di inizio. Ma come fare per ritornare dalla subroutine al punto di partenza, essendo questi punti di salto iniziale non uno solo, ma tanti?

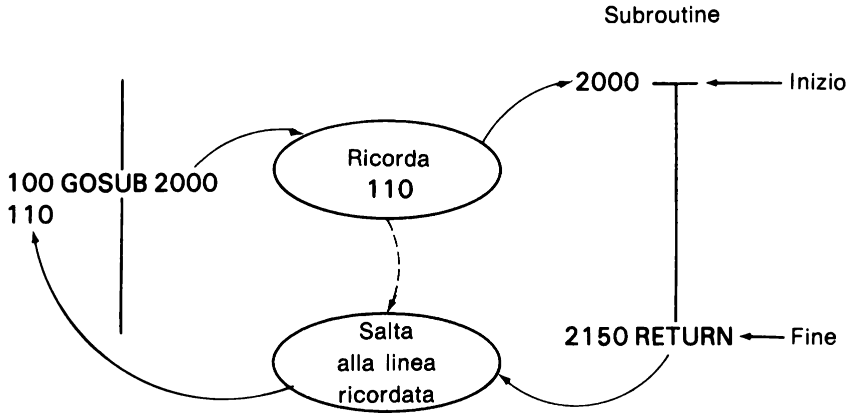
In questo grafico vi mostriamo che, con le GOTO, si può sempre saltare alla subroutine:



Ma al termine della subroutine a quale punto del programma principale si deve ritornare? Il problema viene risolto con una speciale istruzione GOTO che *ricorda* la linea di partenza. Questa istruzione è la GOSUB.

Istruzione GOSUB

L'istruzione GOSUB effettua il salto di partenza come una GOTO, ma in aggiunta ricorda dove ritornare. Nella terminologia dei calcolatori si dice *chiamare* una subroutine. Ecco una illustrazione grafica:



Le subroutine devono terminare con l'istruzione RETURN. Con l'istruzione RETURN il programma prosegue alla istruzione successiva alla GOSUB. Se la GOSUB era l'ultima di una linea, allora il ritorno avviene alla linea successiva.

Le tre istruzioni per inizializzare la variabile A ( ), se convertite in una subroutine, appaiono così:

```
10 REM ** PROGRAMMA PRINCIPALE **
20 REM E' CONSIGLIABILE DIMENSIONARE
30 REM LE VARIABILI DELLE SUBROUTINE
40 REM ALL'INIZIO DEL PROGRAMMA PRINCIPALE.
60 DIM A(99)
70 GOSUB 2000
80 REM VISUALIZZARE QUALCOSA PER
85 REM DIMOSTRARE IL RITORNO DALLA
87 REM SUBROUTINE.
90 PRINT "RITORNATO"
100 END
2000 REM SUBROUTINE
2010 FOR I=0 TO 99
2020 A(I)=I
2030 PRINT A(I)
2040 NEXT I
2050 RETURN
```

Istruzione POP

In alcuni casi può interessarvi che una subroutine non ritorni all'istruzione successiva della GOSUB. Potreste pensare, in questo caso, di usare una istruzione GOTO per uscire dalla subroutine, ma questo causerebbe dei problemi perchè il BASIC continua a ricordare dove la subroutine dovrebbe ritornare. In questi casi potete usare l'i-

struzione POP per annullare il "ricordo" del punto di ritorno. Con l'istruzione POP, infatti, il BASIC annulla il rientro dalla subroutine incontrata per ultima. A questo punto potete allora usare liberamente una GOTO per saltare ad un qualunque punto del programma.

Subroutine nidificate

Le subroutine possono essere nidificate. Questo significa che una subroutine può chiamare un'altra subroutine che a sua volta può chiamare una terza e così via. Per chiamare subroutine nidificate non dovete fare altro che chiamarle con GOSUB, come chiamate la prima, e porre al termine di ognuna RETURN.

Ecco un esempio:

```
10 REM ** PROGRAMMA PRINCIPALE **
20 REM E' CONSIGLIABILE DIMENSIONARE LE
30 REM VARIABILI DELLE SUBROUTINE
40 REM ALL'INIZIO DEL PROGRAMMA PRINCIPALE.
60 DIM A(99)
70 GOSUB 2000
80 REM VISUALIZZARE QUALCOSA PER
85 REM DIMOSTRARE IL RITORNO DALLA
87 REM SUBROUTINE.
90 PRINT "RITORNATO"
100 END
2000 REM SUBROUTINE PRIMO LIVELLO
2010 FOR I=0 TO 99
2020 A(I)=I
2030 GOSUB 3000
2040 NEXT I
2050 RETURN
3000 REM SUBROUTINE SECONDO LIVELLO
3010 PRINT A(I)
3020 RETURN
```

In questo programma l'istruzione PRINT A (I) viene portata fuori dalla subroutine 2000 e inserita nella nuova subroutine 3000.

L'unica limitazione che riguarda le subroutine nidificate concerne la possibilità di una di esse di richiamare se stessa, oppure di richiamare un'altra subroutine che a sua volta ne richiami una già facente parte della nidificazione. Questo concetto, nella terminologia dei calcolatori prende il nome di *ricorsività.* La ricorsività non è permessa nel BASIC dell'Apple II.

Istruzione GOSUB calcolato

Le istruzioni GOTO e GOSUB sono molto simili. L'unica differenza riguarda il fatto che GOSUB ricorda il numero di linea successivo. È quindi ovvio che possa esistere GOSUB calcolato in analogia a GOTO calcolato. L'istruzione GOSUB calcolato per-

mette di saltare ad una, tra più subroutine, a seconda del valore assunto da una espressione numerica. Il ritorno al programma principale avviene dopo l'istruzione RETURN come nel caso della normale istruzione GOSUB.

È possibile nidificare più subroutine con la GOSUB calcolato, proprio come si può fare con la GOSUB normale.

Consideriamo questo esempio in Integer BASIC:

```
>100 GOSUB A*500+2000
>110 REM
```

L'istruzione alla linea 100 è una GOSUB calcolato e quando viene eseguita il programma salta alla subroutine della linea 2000 se A = 0; a quella della linea 2500 se A = 1 e così via. Se il numero di linea calcolato non esiste nel programma, allora viene dato l'errore * * * BAD BRANCH ERR.

Anche nel caso dell'Applesoft l'istruzione GOSUB calcolato è analoga alla GOTO calcolato. Ecco un esempio:

```
]90

]100 ON A GOSUB 1000,500,5000,2300

]110 REM
```

Se A = 1 allora viene richiamata la subroutine 1000, se A = 2 quella della linea 500, se A = 3 quella della linea 5000 e se A = 4 quella della linea 2300. Per qualunque altro valore di A diverso da 1, 2, 3 e 4 il programma prosegue alla linea 110.

ESECUZIONE CONDIZIONALE

Le istruzioni GOTO calcolato e GOSUB calcolato sono due esempi di istruzioni condizionali. Questo significa che l'esatta sequenza di istruzioni, durante l'esecuzione del programma, dipende dai valori assunti da alcune variabili.

Istruzioni IF - THEN

Un'altra istruzione condizionale è l'istruzione IF-THEN. Essa ha la forma generale:

IF *espressione* THEN *istruzione*

Se l'espressione è vera allora *l'istruzione* sarà eseguita. Le espressioni più usate in questo caso sono quelle Booleane e quelle relazionali, ma si possono usare anche

quelle aritmetiche. Tutto ciò dà ai programmi BASIC la possibilità di prendere delle decisioni. Ecco tre semplici esempi:

```
10 IF A=B+5 THEN PRINT MSG$
40 IF CC$="M" THEN IN=0
50 IF Q<14 AND M<M1 THEN GOTO 66
```

L'istruzione alla linea 10 farà eseguire una PRINT se il valore di A è eguale a quello di B più 5. Diversamente la PRINT non sarà eseguita.

Alla linea 40 se CC$ è eguale alla lettera M, allora alla variabile IN viene imposto il valore 0.

Alla linea 50 il programma salta alla linea 66 se Q è minore di 14 e anche M è minore di MI. Diversamente il programma prosegue alla linea successiva.

Per maggiori chiarimenti sul calcolo dell'espressione dopo IF vi rimandiamo alle pagine precedenti.

L'istruzione IF-THEN può essere seguita da altre istruzioni sulla stessa linea, ma il suo comportamento è diverso tra Integer BASIC e Applesoft.

In Integer BASIC solo l'istruzione che segue immediatamente THEN non viene eseguita se l'espressione è falsa. Tutte le altre che seguono sulla stessa linea sono sicuramente eseguite. Questo può essere illustrato così:

```
10 IF V>100 THEN PRINT "AUGURI": GOSUB 2000
20 T=T+V: PRINT T
```

Il messaggio AUGURI viene stampato solo se V è maggiore di 100, ma il salto alla subroutine 2000 viene fatto sempre.

In Applesoft invece tutte le istruzioni che seguono THEN saranno eseguite solo se l'espressione è vera. Diversamente il programma prosegue alla prima istruzione della linea successiva. Nell'esempio di prima quindi il salto alla subroutine 2000 avviene sole se V è maggiore di 100. Diversamente il programma prosegue alla linea 20.

In Applesoft è possibile omettere la parola THEN quando l'istruzione seguente è una GOTO. Queste due istruzioni sono infatti equivalenti:

```
]10 IF MM$=DD$ THEN GOTO 100
```

oppure:

```
]10 IF MM$=DD$ GOTO 100
```

ISTRUZIONI DI INGRESSO E USCITA

Le istruzioni che sovrintendono al trasferimento di dati tra il calcolatore e le periferiche sono chiamate genericamente *istruzioni d'ingresso/uscita*. Le più semplici istruzioni d'ingresso/uscita sono quelle che controllano l'ingresso dei dati dalla tastiera o la loro uscita sullo schermo. Nei paragrafi che seguono descriviamo appunto queste istruzioni, ma rimandiamo ai capitoli 4 e 5 la descrizione delle istruzioni che riguardano i dischi, le cassette e le stampanti. Nel capitolo 6 descriveremo quelle particolari istruzioni che permettono il tracciamento di grafici sullo schermo.

Iniziamo da una istruzione che abbiamo già usato per visualizzare dati sullo schermo: PRINT.

Istruzione PRINT

Precisiamo innanzi tutto che la parola PRINT (in inglese "stampa") deriva dal fatto che i primi calcolatori che operavano in BASIC avevano come periferica di uscita una telescrivente e non un display video perchè allora queste periferiche erano troppo costose. Così il comando più semplice di uscita era appunto "stampa" cioè PRINT.

L'istruzione PRINT permette allora di visualizzare sia dati numerici che stringhe. Per esempio:

```
10 PRINT "TEXT"
```

Per visualizzare un numero potete porlo dopo PRINT oppure assegnarlo ad una variabile:

```
>A=10

>PRINT 5,A
5        10
```

È possibile anche mescolare sia stringhe che numeri. In questo esempio vengono visualizzati sia i nomi UNO, DUE, TRE, QUATTRO e CINQUE che i numeri 1, 2, 3, 4 e 5:

```
10 PRINT "UNO",1,"DUE",2,"TRE",3,"QUATTRO",4,"CINQUE",5
20 END
```

Notate che deve sempre esistere un carattere di separazione tra gli elementi di una PRINT. In questo caso si è scelta la virgola, ma ciò comporta che i dati siano

ampiamente separati tra loro sullo schermo. Se volete eliminare tali spazi dovete usare come carattere separatore il punto e virgola (;):

```
10 PRINT "UNO";1;"DUE";2;"TRE";3;"QUATTRO";4;"CINQUE";5
20 END
```

Provate in modo immediato a dare qualche istruzione PRINT con virgole separatrici oppure con punti e virgole e vedere quindi la differenza.

Una istruzione PRINT termina sempre con un *ritorno del carrello,* cioè con il ritorno a capo del cursore. È possibile però sopprimere il ritorno del carrello ponendo una virgola, o un punto e virgola, in coda alla lista della PRINT. In tal caso l'unica differenza consiste nel fatto che la virgola sposta il cursore di tanti posti come se dovesse visualizzare un altro dato. Per meglio capire questo provate a eseguire il seguente programma:

```
10 PRINT "UNO",1,"DUE",2
20 PRINT "TRE",3,"QUATTRO",4,"CINQUE",5
30 END
```

Aggiungete poi una virgola in coda alla PRINT della linea 10 e rieseguite il programma. Vedrete che i due testi appaiono su una stessa linea dello schermo.

Sostituite ora la virgola con un punto e virgola alla linea 10. Vi accorgerete allora che non vi è più spazio tra il numero 2 e la parola TRE. Cambiando altre virgole in punto e virgola potete togliere altre spaziature.

In questo programma abbiamo fatto visualizzare i numeri ponendoli direttamente nell'istruzione PRINT, ma è possibile anche usare delle variabili numeriche a cui assegnare tali valori numerici. Poniamo allora nella PRINT la variabile A ():

```
5 DIM A(5)
10 FOR I=1 TO 5
20 A(I)=I
30 NEXT I
40 PRINT "UNO";A(1);"DUE";A(2);"TRE";A(3);"QUATTRO";A(4);
"CINQUE";A(5)
50 END
```

In Applesoft potete anche usare delle variabili di stringa per denominare le stringhe e inserire la PRINT in un ciclo FOR-NEXT nel modo seguente:

```
10 DATA "UNO","DUE","TRE","QUATTRO","CINQUE"
20 FOR I=1 TO 5
40 READ N$
50 PRINT N$;I;
60 NEXT I
70 END
```

Istruzione INPUT

Questa istruzione permette l'ingresso di dati dalla tastiera nel calcolatore. Durante la sua esecuzione il calcolatore attende che vengano forniti tutti i dati richiesti e non viene svolta alcuna altra funzione.

Dopo la parola INPUT deve seguire una lista di variabili in corrispondenza alle quali dovrete dare i valori. Ogni variabile determina in corrispondenza il tipo di dato che dovete fornire. Una variabile numerica richiede un dato numerico, mentre una variabile di stringa vuole una stringa in ingresso. Per dimostrarvi l'ingresso di dati numerici provate ad eseguire questo programma:

```
10 INPUT A
20 PRINT A
25 REM IL PROGRAMMA TERMINA SE VIENE DATO 0
30 IF A=0 THEN END
40 GOTO 10
```

Nel momento di esecuzione della INPUT sullo schermo appare un punto interrogativo per avvisarvi che dovete fornire un dato. Appena battete il dato, lo vedete anche apparire sullo schermo. Si dice allora che lo schermo fa da *eco* alla tastiera. Subito dopo il dato viene ancora visualizzato per la presenza di una PRINT alla linea 20.

Una istruzione INPUT può ricevere più di un dato purchè abbia un numero sufficiente di variabili nella sua lista separate tra loro da virgole. Dopo il punto interrogativo dovete quindi battere i valori corrispondenti al tipo delle variabili.

Se battete dati numerici ricordatevi di non porre mai i punti di separazione tra le migliaia e i milioni ecc.; battete cioè 1000 e *non* 1.000!

Questo esempio fa entrare due valori numerici e poi li visualizza:

```
20 INPUT A,B
30 PRINT A,B
35 REM IL PROGRAMMA TERMINA SE VIENE DATO 0
40 IF A=0 OR B=0 THEN END
50 GOTO 20
```

Eseguite allora questo programma. Battete poi un numero dopo il punto interrogativo seguito da una virgola e da un altro numero. Alla fine premete RETURN. Provate ora a dare i dati d'ingresso in modo diverso. Dopo aver battuto il primo numero premete subito RETURN. Il calcolatore vi ricorderà che dovete dare un altro valore con un altro punto interrogativo. Battete il secondo valore e premete ancora RETURN. Quando dovete fare entrare più valori potete allora darli tutti su una linea oppure su linee diverse.

L'istruzione INPUT, in Integer BASIC, opera in modo diverso quando richiede una

stringa. Per prima cosa non visualizza un punto interrogativo. Provate questo esempio:

```
10 DIM A$(19)
20 INPUT A$
30 PRINT A$
35 REM IL PROGRAMMA TERMINA SE ENTRA IL CARATTERE NULLO
40 IF A$="" THEN END
50 GOTO 20
```

Quando eseguite questo programma provate a fare entrare una stringa lunga più di venti caratteri. Vi verrà subito segnalato l'errore * * * STR OVFL ERR e il programma si fermerà. La lunghezza di una stringa in ingresso non può superare la massima lunghezza prevista per la variabile associata nella INPUT.

In Integer BASIC dovete fare entrare le stringhe su una linea separata. Se una istruzione INPUT contiene nella sua lista sia variabili numeriche che di stringa potete battere i dati numerici su una stessa linea, ma dovete usare linee diverse per ogni stringa. Questo è dovuto al fatto che l'Integer BASIC ammette che una virgola possa far parte di una stringa, per cui la virgola non può essere usata come separatore di stringhe. Provate infatti nell'ultimo esempio a fare entrare la stringa CARLO, LUIGI.

Nell'esempio seguente potete vedere che cosa succede se una variabile di stringa fa parte di una INPUT in Integer BASIC. Provate a battere i quattro valori su una stessa linea separati da virgole. Che cosa succede? Provate anche a vedere che cosa succede se fate entrare una virgola o dei numeri come parte di una stringa.

```
10 DIM A$(10),B$(10)
20 INPUT A$,A,B$,B
30 PRINT A$,A,B$,B
35 REM IL PROGRAMMA TERMINA SE ENTRA IL CARATTERE NULLO
40 IF A$="" THEN END
50 GOTO 20
```

Ricordiamo ancora che in Applesoft una variabile reale può ricevere un valore intero. Viceversa se battete un valore reale per una variabile intera, esso viene convertito in intero con un troncamento come vi abbiamo già illustrato nel paragrafo "Espressioni di tipo misto".

Segnali di avviso nelle istruzioni INPUT

Le istruzioni INPUT sono molto delicate. La loro sintassi è molto precisa. Solo una persona esperta di programmazione può capire l'importanza di mettere le virgole nei posti giusti. Pensate come si troverebbe a disagio una persona digiuna di calcolatori se, caricando dei dati, vedesse apparire sullo schermo uno dei nostri strani messaggi di errore.

I programmi professionali hanno quindi la parte, che riguarda l'ingresso dati, scritta in un modo particolare che viene normalmente detto, senza offesa per nessuno, "a prova d'idiota". Questi programmi prevedono praticamente tutti gli errori che un operatore può commettere in ingresso e cercano di porvi rimedio. Nel capitolo 4 vedremo nei dettagli questa speciale tecnica di programmazione.

È possibile però sin da adesso arricchire i nostri programmi con dei messaggi, visualizzati sullo schermo, che permettono di avvisare l'operatore che una operazione di ingresso dati ha inizio. Questo è possibile grazie al fatto che l'istruzione INPUT accetta una stringa subito dopo la parola INPUT che verrà poi visualizzata sullo schermo. Questa stringa, che deve essere posta tra virgolette, può contenere quindi un messaggio di avviso all'operatore.

In Integer BASIC ponete il messaggio di avviso tra virgolette subito dopo la parola INPUT e poi una virgola. Dopo la virgola elencate le variabili d'ingresso. Se queste variabili sono più di una il messaggio appare una sola volta. Se la prima variabile è numerica, subito dopo il messaggio appare un punto interrogativo; se invece la prima variabile è di stringa allora il punto interrogativo non appare. Ecco alcuni esempi:

```
10 DIM A$(10)
20 INPUT "DATE IL NOME E L'ETA'",A$,A
30 PRINT A$;" HA ";A;" ANNI"
35 REM IL PROGRAMMA TERMINA SE ENTRA IL CARATTERE NULLO
40 IF A$="" THEN END
50 GOTO 20
```

Anche in Applesoft il messaggio viene posto dopo la parola INPUT, ma è seguito da un punto e virgola a cui segue la lista delle variabili. La presenza del messaggio sopprime la presenza del punto interrogativo. Se i dati sono forniti su più linee il messaggio appare solo una volta. Ecco alcuni esempi in Applesoft:

```
20 INPUT "DATE IL NOME E L'ETA'";A$,A
30 PRINT A$;" HA ";A;" ANNI"
35 REM IL PROGRAMMA TERMINA SE ENTRA IL CARATTERE NULLO
40 IF A$="" THEN END
50 GOTO 20
```

Istruzione GET

L'istruzione GET, presente solo nell'Applesoft, permette l'ingresso di un solo carattere dalla tastiera. Il carattere non appare contemporaneamente sullo schermo. Il tasto RETURN non è necessario premerlo e il carattere viene preso come stringa o come numero a seconda del tipo di variabile che lo richiama. Provate ad eseguire questo programma:

```
10 GET A$
20 PRINT A$
25 REM IL PROGRAMMA TERMINA SE ENTRA E
30 IF A$="E" THEN END
40 GOTO 10
```

È possibile creare un ciclo di attesa per accettare solo un determinato carattere. Ecco un esempio per ricevere la lettera X:

```
10 GET A$
20 IF A$ <> "X" THEN GOTO 10
30 PRINT A$
40 END
```

Se l'istruzione GET specifica una variabile intera o reale allora il dato in ingresso deve essere una cifra numerica; altrimenti si commette un errore ?SYNTAX ERROR e il programma si ferma. A causa di questo e altri problemi che si hanno con la GET, ricevendo valori numerici, questa istruzione è usata solo per ricevere i caratteri di una stringa.

Normalmente l'istruzione GET è usata in programmi di dialogo tra il calcolatore e l'operatore. Per esempio un programma può attendere che l'operatore batta un carattere per confermare la sua presenza (per esempio Y per yes, cioè sì); ecco come potreste impostare questo "dialogo":

```
10 PRINT "ATTENZIONE, CHIAMATA!"
15 PRINT "BATTERE Y PER CONFERMARE"
20 GET A$
30 IF A$ <> "Y" THEN GOTO 20
40 PRINT "TUTTO BENE!"
50 END
```

Notate che in questo programma non viene visualizzato il carattere battuto alla tastiera. Provate ad inserire le istruzioni per fare l'eco sullo schermo dei caratteri richiamati da una GET.

COME FERMARE E FAR CONTINUARE L'ESECUZIONE DI UN PROGRAMMA

Se volete fermare l'esecuzione di un programma premete contemporaneamente CTRL e C. Se in quel momento il programma stava attendendo dati dalla tastiera per la presenza di una INPUT, dovete premere anche RETURN dopo CTRL-C.

In Integer BASIC vedrete apparire il messaggio STOPPED AT seguito dal numero di linea ove il programma si è fermato. Per continuare l'esecuzione battete il comando CON.

In Applesoft invece il messaggio che vi appare è BREAK IN sempre seguito dal numero di linea. La continuazione del programma si ottiene con CONT.

Il tasto RESET

Per fermare un programma potete anche premere il tasto RESET (per alcune versioni di Apple II invece CTRL-RESET).

Per l'Apple II Plus e l'Apple II con la cartolina Language System, il tasto RESET ha lo stesso effetto di CTRL-C.

Per l'Apple II, senza l'Autostart Monitor, il tasto RESET fa apparire il carattere di pronto del Monitor (*). In questo caso se state usando l'Integer BASIC o il Firmware Applesoft, battete CTRL-C per ritornare al linguaggio che stavate usando prima. Se stavate lavorando con l'Applesoft proveniente da cassetta, battete allora OG per ritornare in Applesoft. Se invece l'Applesoft proveniva da dischetto, battete 3DOG per ritornare in Applesoft.

Se effettuate il ripristino non correttamente, rischiate di perdere il programma BASIC.

Istruzione END

L'istruzione END termina l'esecuzione di un programma. Dopo l'esecuzione di una END non è possibile continuare l'esecuzione di un programma.

Istruzione STOP

In Applesoft è possibile fermare l'esecuzione di un programma con un'altra istruzione: STOP.

Dopo una STOP è possibile continuare l'esecuzione del programma battendo CONT.

Quando la STOP viene eseguita, visualizza sullo schermo il messaggio BREAK IN seguito dal numero di linea ove il programma si è fermato.

Istruzione WAIT

In Applesoft è possibile programmare una pausa durante l'esecuzione mediante l'istruzione WAIT. Questa istruzione ferma l'esecuzione sino a quando ad una certa posizione di memoria venga assegnato un determinato valore. Posizione della memoria e valore sono stabiliti nella WAIT stessa. Potete per esempio fermare il vostro programma sino a che qualcuno prema il bottone sul controllo per giochi numero 1. Ecco come:

```
10 REM ATTESA CHE SIA PREMUTO IL
15 REM TASTO DEL PADDLE 1.
20 PRINT "PREMERE IL TASTO DEL PADDLE 1"
30 WAIT -16286,128
40 PRINT "O.K."
50 END
```

Nel capitolo 8 l'istruzione WAIT sarà descritta ulteriormente.

FUNZIONI

Le funzioni sono un altro elemento del BASIC. Esse assomigliano in parte alle variabili ed in parte alle istruzioni.

Per descrivere che cosa sia una funzione è senz'altro opportuno farne subito un esempio:

```
]10 A=SQR(B)
```

Alla variabile A viene assegnato il valore della radice quadrata di B; SQR ( ) è appunto una funzione. Ecco invece un esempio di funzione di stringa:

```
20 L=LEN(D$)
```

Questa funzione calcola la lunghezza in caratteri di una stringa.

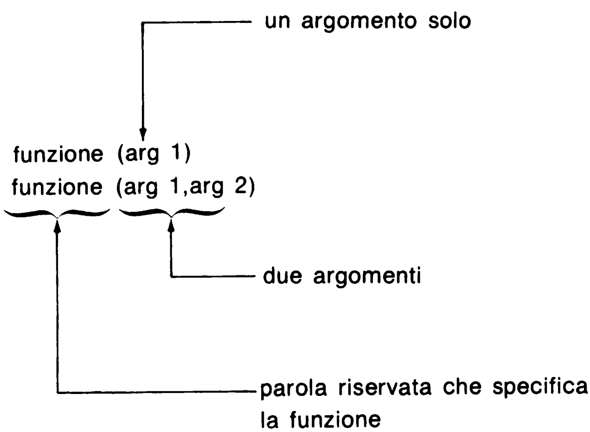
Le funzioni possono essere usate come le variabili o le costanti e inserite ovunque in una istruzione. L'unica eccezione riguarda il fatto che le funzioni non possono essere poste a sinistra del segno di eguale. Potete scrivere A = SQR (B), ma mai SQR (B) = A!

In questo paragrafo descriviamo i concetti fondamentali inerenti le funzioni e come usarle in programmazione. Nel capitolo 8 daremo la descrizione dettagliata di tutte le funzioni disponibili in Integer BASIC e in Applesoft.

Fate attenzione che molte funzioni non sono disponibili in Integer BASIC.

Le funzioni sono sempre definite da una parola riservata seguita da uno o più argomenti posti tra parentesi. Nel caso precedente avevamo SQR per indicare la radice quadrata dell'argomento B; e LEN per indicare la lunghezza della stringa argomento D$.

Generalmente una funzione può avere uno di questi due formati:



Esistono alcune funzioni che operano su tre argomenti.

Ogni argomento può essere una espressione, una variabile o una costante.

Quando l'interprete BASIC incontra una funzione, calcola per prima cosa i suoi argomenti, poi calcola la funzione stessa prima di ogni altra cosa. Cioè le funzioni hanno la massima precedenza. Per esempio in questa istruzione:

```
]10 B=24.7*(SQR(C)+5)-SIN(0.2+D)
```

le funzioni SQR e SIN sono calcolate prima di ogni altro operatore. Supponiamo che SQR (C) = 6.72 e SIN (0.2 + D) = 0.625, l'espressione a destra dell'eguale diviene allora:

24.7 * (6.72 + 5) − 0.625

FUNZIONI NUMERICHE

Riportiamo un elenco di funzioni numeriche che possono essere impiegate sia con l'Integer BASIC che con l'Applesoft:

SGN	Ritorna il segno dell'argomento: +1 per argomento positivo, 0 per argomento zero, −1 per argomento negativo.
ABS	Ritorna il valore assoluto dell'argomento. Se l'argomento è positivo rimane eguale, se negativo viene convertito in positivo.
RND	Genera un numero a caso. (Vedere il capitolo 8).

Le funzioni che seguono sono disponibili solo in Applesoft:

INT	Converte un numero reale in intero.
SQR	Calcola la radice quadrata dell'argomento.
EXP	Calcola la costante *e* elevata alla potenza dell'argomento (e^{arg}).
LOG	Calcola il logaritmo naturale dell'argomento.
SIN	Calcola la funzione trigonometrica seno dell'argomento espresso in radianti.
COS	Calcola la funzione trigonometrica coseno dell'argomento espresso in radianti.
TAN	Calcola la funzione trigonometrica tangente dell'argomento espresso in radianti.
ATN	Ritorna il valore angolare, espresso in radianti, di una tangente.

Uso delle funzioni numeriche

Cercate di usare subito le funzioni nei vostri programmi. Se di alcune non ne comprendete il significato, non preoccupatevene. Forse non potete comprendere l'impiego di quelle trigonometriche come SIN, COS o TAN, ma ora dovete unicamente capire a che cosa serva una funzione nella programmazione.

Vediamo un esempio di funzione numerica:

```
10 A=-234
20 B=SGN(A)
30 PRINT B
40 END
```

Quando eseguite questo programma ottenete –1 poichè –234 è negativo. Per esercizio cambiate la linea 10 con una istruzione INPUT e la linea 40 con GOTO 10. Potete così dare alla tastiera una successione di valori e vedere che cosa significa la funzione SGN.

Ecco un altro esempio più complesso in Applesoft:

```
10 INPUT A,B
20 IF LOG(A) < 0 THEN A=1/A
30 PRINT SQR(A)*EXP(B)
39 REM USARE CTRL-C PER TERMINARE
40 GOTO 10
```

Se conoscete già i logaritmi allora provate a sostituire la funzione LOG della linea 20 con un'altra funzione.

FUNZIONI DI STRINGA

Le funzioni di stringa operano sulle stringhe. Queste funzioni sono per noi molto più importanti di quelle numeriche perchè sono necessarie per potere ben programmare. Vi consigliamo perciò di leggere attentamente le definizioni che seguono.

Le funzioni di stringa, valide sia in Integer BASIC che in Applesoft, sono:

ASC	Converte un carattere di stringa nel codice numerico ASCII.
LEN	Ritorna il numero di caratteri contenuti in una stringa.

Le funzioni di stringa valide solo in Applesoft sono:

STR$	Converte un valore numerico in un equivalente testo di stringa.
VAL	Converte (se possibile) un testo di stringa in un equivalente valore numerico.

CHR$	Converte un codice numerico ASCII nel carattere corrispondente.
LEFT$	Estrae la parte sinistra di una stringa secondo quanto specificato dal suo argomento.
RIGHT$	Estrae la parte destra di una stringa secondo quanto specificato dal suo argomento.
MID$	Estrae la parte intermedia di una stringa in accordo a quanto specificato dai due suoi argomenti.

In altre parole le funzioni di stringa permettono di determinare la lunghezza di una stringa, di estrarne una porzione, di convertire un valore numerico e di codificare e decodificare un codice ASCII. Ecco alcuni esempi:

STR$(14)	Converte 14 in "14"
LEN("ABC")	Ritorna il valore 3, cioè il numero di caratteri di ABC
LEN(A$+B$)	Ritorna la lunghezza delle due stringhe sommate.
LEFT$(ST$,1)	Ritorna il primo carattere a sinistra di ST$.

Sottostringhe dell'Integer BASIC

Sebbene in Integer BASIC non vi siano funzioni che permettono di estrarre parti di una stringa, esiste però una possibilità di farlo come indichiamo in questo esempio:

```
10 DIM A$(20),B$(5)
20 B$=A$(1,4)
```

Nella istruzione 20 B$ è posta eguale ai primi quattro caratteri di A$. Vi potrebbe sembrare di aver posto, con questa istruzione, B$ eguale ad un elemento della variabile con indici A$ ( ). Ma ricordatevi che in Integer BASIC non vi sono variabili con indici di stringhe e tanto meno una variabile con due indici. La notazione che compare alla linea 20 indica una sottostringa. Il primo valore tra parentesi rappresenta la posizione del primo carattere della sottostringa, mentre il secondo valore indica il numero di caratteri della sottostringa.

Concatenamento di stringhe in Integer BASIC

Mediante la funzione LEN è possibile concatenare più stringhe anche in Integer BASIC. Ecco come:

```
10  DIM A$(10),B$(10),C$(10)
20  A$="WIND"
30  B$="PIPE"
40  C$="LINE"
50  A$( LEN(A$)+1)=B$
60  PRINT A$
70  B$( LEN(B$)+1)=C$
80  PRINT B$
90  END

>RUN
WINDPIPE
PIPELINE
```

FUNZIONI DI SISTEMA

Per rendere completo questo capitolo riportiamo a questo punto le funzioni di sistema anche se non le potrete usare fino quando non sarete esperti programmatori:

PEEK	Legge il contenuto di una posizione di memoria.
FRE	Ritorna lo spazio di memoria di lettura/scrittura disponibile; il valore viene dato in byte. Questa funzione non è disponibile in Integer BASIC.
USR	Permette il trasferimento ad un programma scritto nel linguaggio di macchina. Non è disponibile in Integer BASIC.

FUNZIONI DEFINITE DALL'UTENTE

Oltre alle funzioni che abbiamo appena descritto e che fanno parte del BASIC, è possibile in Applesoft usare delle funzioni definite dallo stesso programmatore. Queste nuove funzioni non possono essere però funzioni di stringa. Per definire una funzione si usa l'istruzione DEF FN. Per richiamare tale funzione dovete usare l'istruzione FN. Ecco un esempio:

```
10 DEF FN P(X) = 100 * X
20 INPUT A
30 PRINT A FN P(A)
35 REM USARE CTRL-C PER TERMINARE
40 GOTO 20
```

Dopo la parola riservata FN segue il nome identificatore della funzione. Questo nome ha le stesse caratteristiche dei nomi delle variabili. Nell'esempio l'identificatore è P e per richiamare la funzione si usa FNP. Se fosse stato AB avremmo avuto FNAB.

La funzione viene definita mediante l'espressione aritmetica posta a destra del segno eguale. Quando la funzione viene richiamata mediante la FN, l'espressione arit-

metica viene calcolata (usando i valori correnti per le variabili contenute in essa). Il risultato di questa espressione viene associato alla funzione e usato come un qualunque valore numerico.

Nell'istruzione DEF FN dopo l'identificatore deve essere presente una variabile posta tra parentesi. Questa variabile prende il nome di *variabile locale* in quanto non ha effetto fuori dell'istruzione DEF FN. Questo nome di variabile può essere infatti usato in altri punti del programma senza influenzare la funzione e viceversa.

Analogamente dopo la parola FN e l'identificatore deve comparire tra parentesi una costante numerica o una variabile o una espressione. Quando l'Applesoft incontra una FN assegna, quanto posto tra parentesi, alla variabile locale della corrispondente DEF FN. (L'eventuale valore della variabile, che porta lo stesso nome della variabile locale, rimane immutato). Se la variabile locale è presente nell'espressione di definizione della funzione, l'Applesoft le assegna il valore più recente.

FUNZIONI NIDIFICATE

L'argomento di una funzione può essere una espressione che a sua volta può contenere altre funzioni. In altre parole le funzioni possono essere nidificate. Ecco un esempio:

```
10 INPUT A
20 PRINT SGN( ABS (A) )
25 REM USARE CTRL-C PER TERMINARE
30 GOTO 10
40 END
```

Provate con delle istruzioni PRINT in modo immediato a fare uso di funzioni numeriche o di stringa.

CAPITOLO 4

PROGRAMMAZIONE BASIC AVANZATA

Questo capitolo è la continuazione del capitolo 3. Molte cose che nel capitolo precedente vi sono state parzialmente accennate, vengono ora approfondite così da permettervi di scrivere dei programmi ad un buon livello di programmazione.

ACCESSO DIRETTO E CONTROLLO

Vi sono alcune istruzioni che permettono un accesso diretto al calcolatore Apple II. Con queste istruzioni potete operare con il controllo dei giochi, far funzionare l'altoparlante e fare un uso completo di tutte le periferiche.

MEMORIA E INDIRIZZAMENTO

Il calcolatore Apple II può avere sino a 65.536 posizioni di memoria indirizzabili ognuna delle quali può contenere un numero compreso tra 0 e 255. (Questo limite superiore è pari a 2^8). Tutti i programmi e i dati sono sempre convertiti in una sequenza di numeri che vengono memorizzati.

Per ognuna delle istruzioni, che in seguito illustreremo, dovete sempre indicare un indirizzo di memoria. Questo indirizzo può essere specificato come costante, come variabile o come espressione, ma in ogni caso deve portare ad un indirizzo valido.

Ogni posizione di memoria può essere individuata da due indirizzi. Il primo è positivo e compreso tra 0 e 65535. Il secondo è negativo ed è ottenuto dal primo sottraendo il valore 65536. Per esempio −32767 e 32768 si riferiscono alla stessa posizione di memoria. Così anche −1 e 65535 danno luogo allo stesso indirizzamento.

Se tenete presente che il più grande numero intero in Integer BASIC è 32767, potete capire l'utilità di usare i valori negativi per indirizzare le posizioni alte della memoria.

Se in Applesoft usate un numero reale per indicare un indirizzo di memoria, esso sarà convertito in intero.

PEEK e POKE

La funzione PEEK permette di leggere il contenuto di una qualunque posizione di memoria dell'Apple II. Consideriamo per esempio:

```
10 A = PEEK(200)
```

Il contenuto della posizione 200 di memoria viene assegnato alla variabile A.

L'istruzione POKE *inserisce* cioè scrive un certo valore in una determinata posizione di memoria. Per esempio:

```
20 POKE 8000,A
```

Il valore della variabile A viene inserito nella posizione 8000. Il valore che si vuole scrivere può essere una costante, una variabile o una espressione purchè porti ad un risultato compreso tra 0 e 255.

La funzione PEEK può essere applicata sia alla memoria di lettura/scrittura che a quella di sola lettura (RAM e ROM); la funzione POKE solo a quella di lettura/scrittura (RAM). Tutto ciò è facilmente comprensibile perchè la memoria di sola lettura ROM non può essere scritta dal programmatore, ma può essere solamente letta.

Istruzione CALL

Mediante l'istruzione CALL è possibile trasferire il controllo dal BASIC ad una subroutine o ad un programma scritto in assembler. Ecco un esempio:

```
100 CALL A1
```

Il controllo viene trasferito alla posizione di memoria indicata dalla variabile A1.

Le subroutine scritte in assembler possono essere quelle che risiedono sempre nella memoria dell'Apple II (cioè nella ROM) oppure quelle che voi stessi scrivete. Il Monitor ha per esempio una subroutine che pulisce lo schermo. Nella Appendice D sono riportate tutte le subroutine intrinseche dell'Apple II. Per maggiori spiegazioni sul Monitor e sul linguaggio assembler vi rimandiamo al capitolo 7.

Istruzioni HIMEM: e LOMEM:

La memoria dell'Apple II è usata in vari modi. Una parte è occupata dal vostro programma BASIC, un'altra dalle variabili del vostro programma, un'altra ancora dai programmi per la gestione dei dischi (ovviamente se li avete collegati al calcolatore) e così avanti per altri scopi. Una parte della memoria è anche occupata dai grafici come vedremo meglio nel capitolo 6.

Le istruzioni HIMEM: e LOMEM: permettono di riservare una parte della memoria per programmi in assembler o per i grafici ad alta risoluzione.

Ecco due esempi di queste istruzioni:

```
50  HIMEM: 38400
60  LOMEM: 12291
```

Normalmente i limiti superiore e inferiore sono posti automaticamente dall'Apple II a seconda delle posizioni di memoria disponibili per i programmi BASIC. Alcune volte però dovete spostare questi limiti come nel caso che scriviate programmi in assembler o disegniate grafici ad alta risoluzione. (Vedere i capitoli 6 e 7 e anche l'Appendice G).

IMPIEGO DELLE PERIFERICHE

Normalmente il vostro calcolatore Apple II comunica con l'esterno mediante due periferiche privilegiate: la tastiera per l'ingresso dei dati e lo schermo per la loro uscita.

È possibile però eseguire operazioni d'ingresso/uscita anche con altre periferiche. Tali periferiche possono essere:

- Un registratore a cassette per memorizzarvi programmi (e anche dati).
- Una unità a disco per memorizzarvi sia programmi che dati.
- Una stampante per listare sia programmi che dati.
- Una tavoletta grafica per l'ingresso di disegni.
- Un canale di comunicazione con altri calcolatori.

Il registratore a cassette è collegato all'Apple II mediante due connettori «jack».

Tutte le altre periferiche sono collegate mediante cartoline inserite negli slot interni del calcolatore. Queste cartoline possono essere chiamate anche *controllori* oppure *interfacce*.

Quando in un programma dovete richiamare una periferica, userete come sua identificazione proprio il numero fisico dello slot a cui è collegata (da 0 a 7). Lo schermo e la tastiera hanno sempre il numero 0.

Istruzioni PR# e IN#

Dopo quanto si è detto, è comprensibile che si possa cambiare a programma la periferica di ingresso come anche quella di uscita. Con l'istruzione PR# cambiate la periferica di uscita; con IN# quella d'ingresso. Questa istruzione in modo immediato seleziona lo schermo come organo di uscita:

```
PR#0
```

Se nel vostro calcolatore non avete caricato il DOS, potete usare le istruzioni PR# e IN# sia in modo immediato che differito. Se invece state lavorando con il DOS l'uso di queste due istruzioni, in modo differito, è un po' più complicato. In questo caso dovete usare una PRINT che visualizzi un carattere CTRL-D seguito immediatamente da PR# o IN#. In questo esempio si suppone che una stampante sia collegata allo slot 1; con il comando PR#1 si commuta l'uscita dallo schermo alla stampante:

```
100 D$="" : REM CTRL-D
110 PRINT D$;"PR#1": REM SELEZIONA SLOT 1 STAMPANTE
```

Se il DOS non fosse presente, sarebbe stato sufficiente invece porre:

```
110 PR#1 : REM SELEZIONA SLOT 1 STAMPANTE
```

Notate infine che le due istruzioni PR# e IN# richiedono che sia definito un argomento compreso tra 0 e 7. Se ponete un argomento diverso commettete un errore che porta a risultati imprevedibili. Se invece l'argomento è compreso tra 0 e 7, ma non vi è alcuna cartolina nello slot corrispondente, il calcolatore entra in uno stato di indecisione e l'unico modo per uscirne è premere RESET.

USCITA E INGRESSO DATI

Le parti di un programma concernenti l'ingresso e l'uscita dei dati sono sicuramente le più delicate e le più impegnative nella loro stesura.

È ovvio pensare che un programma professionale non possa ricevere dati in ingresso solo con qualche istruzione INPUT. Inoltre quali possono essere le conseguenze se l'operatore batte un tasto sbagliato o, ancora peggio, si accorge di avere dato un valore errato dopo altri dati? Un programma professionale deve tenere presente che l'operatore può umanamente commettere errori e, nei limiti del possibile, cercare di porvi rimedio.

Analogamente i dati in uscita devono presentarsi in forma chiara e leggibile. Di conseguenza non sarà sufficiente porre qualche istruzione PRINT, ma sarà necessario tabularli in maniera molto intelleggibile.

Fortunatamente il BASIC dell'Apple II contiene tutte le istruzioni necessarie per fare una buona programmazione di ingresso e uscita dei dati. Analizzeremo ora queste istruzioni prima di descrivere delle buone tecniche di ingresso/uscita dati.

ULTERIORI CHIARIMENTI SULLA ISTRUZIONE PRINT

Normalmente una istruzione PRINT termina con un invio del cursore a capo (cioè facendo un *ritorno del carrello*). Di conseguenza una eventuale PRINT successiva i-

nizia a visualizzare sulla prima colonna della riga successiva.

Osservate questo esempio in cui si visualizzano, sempre a capo, 20 caratteri W:

```
200 C$ = "W"
210  FOR I = 1 TO 20
220  PRINT C$
230  NEXT I
240  PRINT "UFFA!"
250  END

]RUN
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
W
UFFA!
```

Uso del punto e virgola

Un carattere di punto e virgola (;), posto dopo una variabile in una PRINT, fa iniziare la visualizzazione della variabile seguente alla posizione subito successiva. Un punto e virgola posto dopo l'ultima variabile, di una istruzione PRINT, sopprime il ritorno del carrello.

In questo esempio vengono visualizzati 800 caratteri W su 20 righe e 40 colonne:

```
200 C$ = "W"
210  FOR I = 1 TO 800
220  PRINT C$;
230  NEXT I
240  PRINT "UFFA!"
250  END
```

```
]RUN
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
UFFA!
```

La variabile I del ciclo FOR-NEXT è usata come contatore delle 800 W. Durante la prima esecuzione del ciclo la PRINT inizia a visualizzare sulla prima colonna, poi non va a capo, per la presenza del punto e virgola, e continua così a porre le W sulla prima riga sino quando la riga è riempita. A questo punto il cursore va necessariamente a capo anche se la PRINT continua a visualizzare sempre sulla stessa riga. Viene allora riempita la seconda riga e così via per tutte le 800 W.

La parola UFFA! sembra scritta a capo solamente perchè l'ultima W era posta sull'ultima colonna della riga precedente. Provate infatti a cambiare l'estremo 800 con 780 e vedrete che l'esclamazione UFFA! rimane sulla stessa riga delle ultime W:

```
200 C$ = "W"
210  FOR I = 1 TO 780
220  PRINT C$;
230  NEXT I
240  PRINT "UFFA!"
250  END

]RUN
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
```

```
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWUFFA!
```

Il punto e virgola permette quindi di visualizzare le stringhe come se fossero concatenate. Anche i dati numerici appaiono senza spazi intermedi, salvo che sia presente il segno meno di un valore negativo.

Proviamo allora per esempio a porre un valore numerico positivo al posto del carattere W:

```
200 C = 5
210  FOR I = 1 TO 800
220  PRINT C;
230  NEXT I
240  PRINT "UFFA!"
250  END
```

Il numero 5 viene così visualizzato:

```
]RUN
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
UFFA!
```

Cambiamo ora C in –1:

```
200 C =  - 1
210  FOR I = 1 TO 800
220  PRINT C;
230  NEXT I
240  PRINT "UFFA!"
250  END

]RUN
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
UFFA!
```

Come mai in questo caso le righe che appaiono sullo schermo sono molto più nu-

merose? La presenza del segno meno ha infatti richiesto più spazio per cui i caratteri effettivamente visualizzati sono raddoppiati.

Se il valore di C fosse stato un numero con più cifre si avrebbe avuto uno scorrimento ("scroll") dell'immagine sullo schermo. Se poniamo infatti C=2001 dobbiamo ridurre l'estremo del ciclo a 800/4=200 per evitare lo scroll:

```
200 C = 2001
210  FOR I = 1 TO 200
220  PRINT C;
230  NEXT I
240  PRINT "UFFA!"
250  END

]RUN
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
2001200120012001200120012001200120012001
UFFA!
```

Se ora provassimo a porre un numero come 201 per C, troveremmo che al termine delle righe i numeri verrebbero spezzati perchè non vi è nulla nella PRINT che separa i valori 201 tra di loro, ma solo la fine della riga manda il cursore a capo.

Uso della virgola

La presenza di una virgola dopo una variabile fa si che la variabile successiva venga visualizzata dopo un determinato intervallo di tabulazione. Questa tabulazione è diversa tra Integer BASIC e Applesoft.

In Integer BASIC esistono cinque punti di tabulazione sullo schermo posti alle co-

lonne 1, 9, 17, 25 e 33. La spaziatura tra di loro è di otto posti. Ecco un esempio in Integer BASIC:

```
>NEW

> 200 C=123
> 210 FOR I=1 TO 20
> 220 PRINT C,
> 230 NEXT I
> 240 PRINT "UFFA!"
> 250 END

>RUN
123      123      123      123      123
123      123      123      123      123
123      123      123      123      123
123      123      123      123      123
UFFA!
```

Se la posizione precedente a quella di un punto di tabulazione, escluso il primo, non è vuota quel punto di tabulazione non viene attivato. Cioè per tabulare in colonna 9 non vi deve essere scritto niente in colonna 8; e così via per gli altri punti di tabulazione. A questo proposito osservate l'esempio seguente che visualizza stringhe:

```
>10 REM LE VARIABILI DI STRINGA DEVONO ESSERE DIMENSIONATE
>20 DIM C$(8)
>200 C$="12345678"
>210 FOR I=1 TO 10
>220 PRINT C$,
>230 NEXT I
>240 PRINT "UFFA!"
>250 END

>RUN
12345678                12345678                12345678
        12345678                12345678
12345678                12345678                12345678
        12345678                12345678
UFFA!
```

Se cambiate il valore di C$ in "1234567" vedrete che tutti e cinque i punti di tabulazione saranno utilizzati.

In Applesoft esistono alcune particolari condizioni per effettuare la tabulazione come indicato nella Figura 4.1.

Per vedere come funziona la tabulazione in Applesoft, cambiate il punto e virgola in virgola nel programma con C=2001 e riducete l'estremo del ciclo FOR-NEXT da 200 a 3 * 20 = 60.

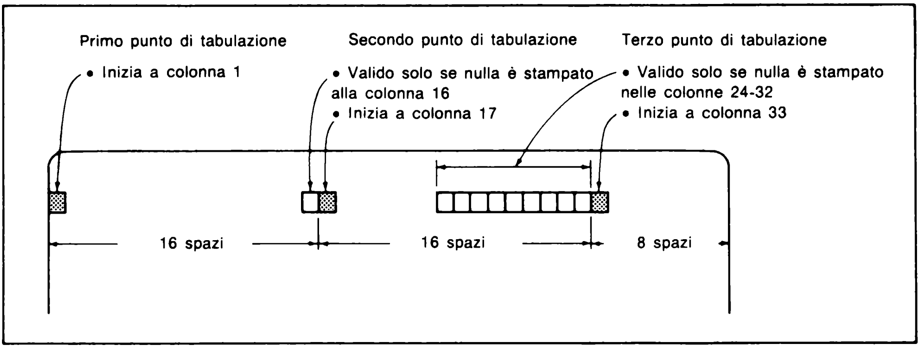


Figura 4.1 — Punti di tabulazione in Applesoft (PRINT con virgole).

```
200 C = 2001
210  FOR I = 1 TO 60
220  PRINT C,
230  NEXT I
240  PRINT "UFFA!"
250  END

]RUN
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
2001            2001            2001
UFFA!
```

Le virgole possono essere usate anche con le stringhe. Provate infatti a porre in esecuzione il seguente programma:

```
100 A$ = "TRE"
110 B$ = "GATTI"
120 C$ = "CECHI"
210  FOR I = 1 TO 20
220  PRINT A$,B$,C$
230  NEXT I
240  PRINT "UFFA!"
250  END

]RUN
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
TRE               GATTI             CECHI
UFFA!
```

Sempre come esercizio provate a porre B$ = GATTACCI. Vedrete, in accordo con la Figura 4.1, che il terzo punto di tabulazione non viene più usato perchè la parola GATTACCI cade nell'intervallo di posizioni da 24 a 32.

FUNZIONI DI FORMATO DELLA PRINT

Usiamo la parola *formato* per indicare come viene presentato un testo in uscita o come deve essere dato in ingresso. Abbiamo già visto per esempio che le virgole e i punti e virgola modificano il formato di una PRINT. Esistono però problemi di formato molto più complessi. Supponete per esempio di voler porre un titolo in mezzo ad una linea dello schermo.

In questo programma, scritto in Integer BASIC, vengono usati i punti di tabulazione per portare un titolo in mezzo alla linea e viene richiamata una subroutine con CALL per portare il cursore in alto a sinistra (vedere l'Appendice D):

```
>NEW

>5 REM DIMENSIONARE LE VARIABILI DI STRINGA
>10 DIM HD$(30)
>500 REM PULISCE LO SCHERMO E
>510 REM PORTA IL CURSORE NELL'
>520 REM ANGOLO IN ALTO A SINISTRA
>530 CALL -936
>540 HD$="SEGRETISSIMO"
>550 REM TABULAZIONE NEL
>555 REM CENTRO DELLO SCHERMO
>560 PRINT "","",HD$
>570 END

>RUN

                    SEGRETISSIMO
```

Come potete vedere il titolo è in mezzo allo schermo. Se fosse stato più lungo come RAPPORTO TRIMESTRALE sarebbe stato allora troppo a destra. Per ovviare a questo si può togliere dalla PRINT della linea 560 una virgola, e la stringa nulla seguente, così che il titolo inizi al secondo punto di tabulazione invece che al terzo.

Anche in Applesoft si può procedere nello stesso modo, ma è possibile anche fare uso di funzioni speciali che sono SPC, TAB e POS.

La funzione SPC

La funzione SPC ( ) permette di spostare il cursore di un numero di passi indicati dal suo argomento. SPC ( ) deve essere inserita nella lista di una PRINT come una qualunque variabile. Nell'esempio precedente avremmo potuto posizionare il titolo SEGRETISSIMO In questo modo:

```
]100 REM PULISCE LO SCHERMO
]110 HOME
]120 REM SPOSTAMENTO SULLO SCHERMO
]130 PRINT SPC(15);"SEGRETISSIMO"
]140 END

]RUN

                   SEGRETISSIMO
```

Notate che dopo SPC compare un punto e virgola. Se vi fosse stata invece una virgola avremmo avuto anche l'effetto di tale virgola, cioè di saltare al successivo punto di tabulazione.

In altre parole la funzione SPC permette di spostare il cursore di quante posizioni si vuole su una linea.

La funzione TAB

La funzione TAB opera come la tabulazione di una normale macchina da scrivere. Supponiamo di dover stampare o visualizzare dei dati in colonna. È necessario allora calcolare la posizione in cui iniziano le singole colonne di dati. Per fare questo lavoro vi può essere utile usare un foglio quadrettato come quello riportato nell'Appendice L. Nella Figura 4.2, per esempio, le colonne iniziano alle posizioni 1, 15 e 31. Il probelma può essere risolto con l'inserimento di spazi vuoti tra un dato e il successivo, ma è anche possibile usare la funzione TAB inserita nell'istruzione PRINT.

Per ogni linea di testo della Figura 4.2 potremmo pensare di inserire degli spazi in questo modo:

```
]10 ?"JONES,P.J      431-25-6277      1420.00"
```

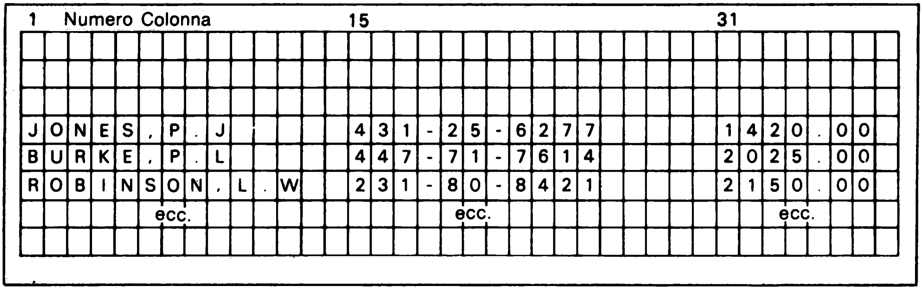


Figura 4.2 – Determinazione delle posizioni dei caratteri sullo schermo.

Oppure usare la funzione SPC e scrivere invece cosi:

```
]10 ?"JONES,P.J";SPC(7);"431-25-6277";SP
C(5);"1420.00"
```

Con TAB è invece possibile posizionare tutti i dati senza dovere contare ogni volta gli spazi intermedi:

```
]10 ?"JONES,P.J";TAB(16);"431-25-6277";T
AB(32);"1420.00"
```

Determinazione della posizione del cursore (orizzontale)

L'ultima delle funzioni di formato dell'Applesoft è POS. POS indica la posizione attuale del cursore, cioè il numero di colonna. Per richiamare la funzione POS è necessario porre un argomento fittizio nullo tra parentesi: POS (O).

Ecco un esempio:

```
]?"POSIZIONE DEL CURSORE: ";POS(O)
```

Se eseguite questa istruzione in modo immediato, ottenete:

```
]?"POSIZIONE DEL CURSORE: ";POS(O)

POSIZIONE DEL CURSORE: 24
```

Se cambiate la stringa tra virgolette, vedrete apparire un nuovo numero di posizione del cursore.

In Integer BASIC potete simulare la funzione POS (O) con PEEK (36). Ecco infatti lo stesso esempio in Integer BASIC:

```
>PRINT "POSIZIONE DEL CURSORE: ";PEEK(36)

POSIZIONE DEL CURSORE: 24
```

Notate che le colonne dello schermo sono numerate da 0 a 39 per POS, PEEK (36) e PRINT SPC. Sono invece numerate da 1 a 40 per PRINT TAB e per due nuove funzioni che vedremo in seguito HTAB e TAB (per l'Integer BASIC).

Determinazione della posizione del cursore (verticale)

La funzione PEEK (37) ritorna il numero di riga su cui si trova il cursore.

Le righe sono numerate da 0 a 23 per PEEK (37) e da 1 a 24 per VTAB che vedremo in seguito.

CONTROLLO DEL CURSORE ED EFFETTI SPECIALI DEL VIDEO

Mediante le istruzioni FLASH, INVERSE, SPEED, NORMAL, HOME, VTAB e HTAB/TAB è possibile usare lo schermo video con maggiore versatilità. Molte di queste istruzioni sono però disponibili solo con l'Applesoft. Nel capitolo 6 vedremo inoltre alcune istruzioni grafiche.

Posizionamento del cursore

Abbiamo già visto come sia possibile controllare la posizione del cursore usando, come separatori in una istruzione PRINT, o le virgole o i punti e virgola. In Applesoft è possibile usare inoltre le funzioni SPC, TAB e POS.

Rimozione dei dati dallo schermo video

Per rimuovere tutti i dati dallo schermo, cioè per pulirlo, e portare il cursore nella posizione "home" di riposo (in alto a sinistra) potete usare l'istruzione CALL-936 oppure in Applesoft l'istruzione HOME. Provate infatti ad usare queste due istruzioni la prima in Integer BASIC e la seconda in Applesoft:

```
>CALL -936

]HOME
```

Posizionamento orizzontale e verticale

Esistono due istruzioni che vi permettono di portare il cursore in un punto qualunque dello schermo. VTAB muove il cursore verticalmente mentre HTAB (o TAB in Integer BASIC) lo muove orizzontalmente. VTAB vuole che si specifichi il numero di linea (1 per la linea più in alto e 24 per quella più in basso); HTAB vuole che si specifichi il numero di colonna (1 per la colonna più a sinistra e 40 per quella più a destra).

Nel seguente esempio, in Applesoft, viene posizionato un asterisco in un punto prefissato dello schermo. Se lavorate in Integer BASIC usate TAB invece di HTAB alle linee 90 e 120:

```
90  HTAB 1: VTAB (1)
100  INPUT "ROW?";R
110  INPUT "COLUMN?";C
120  VTAB R: HTAB C
130  PRINT "*";
140  GOTO 90
```

Le istruzioni INVERSE E NORMAL

L'istruzione INVERSE permette di invertire la parte bianca di un carattere sullo schermo con il nero del fondo (in altre parole si creano caratteri come se fossero il negativo di una fotografia). Dopo l'esecuzione dell'istruzione INVERSE tutti i caratteri visualizzati dalle PRINT appaiono in *modo inverso.* I caratteri battuti alla tastiera continuano invece ad apparire in *modo normale.*

Per ritornare in modo normale dovete dare l'istruzione NORMAL.

Provate in pratica come funzionano queste due istruzioni, battendo i seguenti programmi (le zone ombreggiate sono quelle in modo inverso):

```
]INVERSE

]?"BLACK ON WHITE"
BLACK ON WHITE

]NORMAL

]?"WHITE ON BLACK"
WHITE ON BLACK
```

Le istruzioni INVERSE e NORMAL non sono disponibili in Integer BASIC.

L'istruzione FLASH

Questa istruzione fa apparire alternativamente i caratteri sullo schermo in modo normale e in modo inverso. Anche in questo caso l'istruzione NORMAL fa ritornare al modo normale. I caratteri battuti alla tastiera non sono influenzati dalla FLASH.

Ecco un esempio (le zone ombreggiate sono quelle che lampeggiano):

```
]FLASH
]?"FLASH IN THE PAN"
FLASH IN THE PAN

]NORMAL

]?"STEADY AS RAIN"
STEADY AS RAIN
```

In Integer BASIC non esiste l'istruzione FLASH.

L'istruzione SPEED

L'istruzione SPEED permette di diminuire la velocità con cui i caratteri sono inviati allo schermo televisivo.

Provate infatti a vedere che cosa succede eseguendo questo programma:

```
]100  INPUT "SPEED = ";SP

]110  SPEED = SP

]120  FOR CT = 1 TO 3

]130  PRINT "HIC"

]140  NEXT

]150  PRINT "HICCUP"

]160  SPEED = 255

]170  END
```

La velocità massima è SPEED = 255 mentre quella minima è SPEED = 0.

È importante precisare che l'istruzione SPEED influisce anche sulla velocità con cui vengono mandati i caratteri alle altre periferiche.

In Integer BASIC non esiste l'istruzione SPEED.

FINESTRE SULLO SCHERMO

Normalmente il calcolatore Apple II usa 24 linee e 40 colonne sullo schermo. È possibile però, mediante alcune istruzioni POKE, cambiare l'ampiezza di questa finestra. In Tabella 4.1 sono riportate le quattro posizioni di memoria che contengono i parametri di controllo della finestra sullo schermo.

Fate molta attenzione a impostare i giusti valori che riguardano una nuova finestra, perchè altrimenti potreste incorrere in notevoli errori.

Tabella 4.1 — Posizioni di memoria dei parametri della finestra.

Indirizzo	Agisce su	Valori consentiti
32	Margine sinistro	da 0 a 39
33	Larghezza	da 1 a 40 meno il margine sinistro
34	Linea superiore	da 0 alla linea inferiore
35	Linea inferiore	linea superiore meno 24

Il programma che riportiamo qui di seguito apre una finestra in mezzo allo schermo alta due righe per l'ingresso di dati numerici. Per apprezzare l'utilità di questa tecnica provate a dare in ingresso valori non-numerici ed osservate che cosa succede con i messaggi di errore.

Notate inoltre che la presenza della finestra non pulisce il resto dello schermo, ne porta il cursore al suo interno. Per ottenere questo dovete porre altre istruzioni BASIC.

```
1000  REM  SET WINDOW TOP LINE, WIDTH, LEFT MARG., & BOTTOM
 LINE
1010 T = 10:W = 20:LM = 11:B = 13
1020  REM  CLEAR SCREEN
1030  CALL  - 936
1040  REM  SET TEXT WINDOW FOR INPUT
1050  GOSUB 3200
1060  REM  SURROUND INPUT WINDOW WITH ASTERISKS
1070  GOSUB 3000
1080  REM  MOVE CURSOR INSIDE WINDOW; INPUT
1090  VTAB T + 2
1100  INPUT M1
```

```
1110  REM  RESET WINDOW TO FULL SCREEN
1120  GOSUB 3300
1130  REM   MOVE CURSOR TO BOTTOM LINE
1140  VTAB 23
1150  END
2990  REM  SURROUND WINDOW WITH ASTERISKS
3000  VTAB T + 1
3010  GOSUB 3100
3020  VTAB B + 1
3030  GOSUB 3100
3040  RETURN
3090  REM  PRINT ASTERISKS
3100  FOR I = 1 TO W
3110  PRINT "*";
3120  NEXT I
3130  RETURN
3190  REM  SET INPUT TEXT WINDOW
3200  POKE 32,T
3210  POKE 33,W
3220  POKE 34,LM
3230  POKE 35,B
3240  RETURN
3290  REM  SET FULL-SCREEN WINDOW
3300  POKE 32,0
3310  POKE 33,40
3320  POKE 34,0
3330  POKE 35,24
3340  RETURN
```

LA FUNZIONE CHR$: PROGRAMMAZIONE DEI CARATTERI IN ASCII

Nel capitolo precedente abbiamo visto alcuni caratteri non visibili come CTRL-G che fa suonare il "beep". Ci sono però altri caratteri (sia visibili che non visibili) che non possono essere battuti direttamente alla tastiera come "[" e "\". Con la funzione CHR$ è possibile invece generarli indirettamente (solo in Applesoft).

Per comprendere come funziona CHR$ dobbiamo prima vedere come i caratteri sono memorizzati nella memoria del calcolatore Apple II. Siccome la memoria può memorizzare solo numeri, allora i caratteri sono convertiti in numeri.

L'Apple II codifica i caratteri con lo stesso codice usato da tutti gli altri microcalcolatori: il codice ASCII. ASCII è l'acronimo di "American Standard Code for Information Interchange". Per esempio il codice ASCII di A è 65, di B è 66 e così via. La tabella di tutto il codice ASCII è riportata nell'Appendice I. Ogni volta che l'Apple II incontra una stringa la interpreta come una successione di numeri codificati in ASCII.

In Applesoft quando non potete battere un tasto per inserirlo in una stringa potete sempre usare la sua codifica ASCII mediante la funzione CHR$. Per esempio per

creare il simbolo $, cercate il suo valore ASCII nell'Appendice I. Inseritelo quindi come argomento di CHR$ come segue:

```
]PRINT CHR$(36)
$
```

Provate a generare tutti i caratteri ASCII da 0 a 255.

Potete usare la funzione CHR$ in una normale istruzione PRINT nel modo seguente:

```
]?CHR$(34);"ZOUNDS!";CHR$(34)
"ZOUNDS!"
```

La funzione CHR$ vi permette quindi di inserire in una stringa caratteri che diversamente non potreste mai usare come il ritorno del carrello e le virgolette!

PROGRAMMAZIONE DELL'INGRESSO DEI DATI

Un buon programma deve essere in grado di ridurre al minimo le possibilità di errore nell'ingresso dei dati, come anche di rendere facile per l'operatore la loro eventuale correzione.

Quando dovete caricare un insieme di dati, cercate di raggrupparli in *blocchi funzionali*. Ogni singolo blocco dovrà poi essere elaborato separatamente. Evitate cioè di processare un dato alla volta appena esso viene battuto alla tastiera.

Per esempio un programma per la gestione di indirizzi postali richiede in ingresso sia nominativi che indirizzi. Ogni nominativo/indirizzo deve essere trattato come un singolo blocco funzionale. L'operatore deve essere quindi libero di fornire sia il nominativo che l'indirizzo e di controllare se li ha battuti esattamente; solo in un secondo tempo tali dati possono essere ricevuti ed elaborati, tutti e due assieme, dal calcolatore. A questo punto il calcolatore passa alla richiesta successiva di un altro nominativo/indirizzo.

Nel nostro esempio sarebbe poco professionale richiedere un nominativo e processarlo subito; poi chiedere la via e processarla subito e così via anche per la città e per lo stato. Desideriamo invece far capire l'importanza di organizzare, la parte di un programma che riguarda l'ingresso dei dati, in modo tale che i dati rimangano sullo schermo tutto il tempo necessario per controllarli, modificarli e/o correggerli prima di inviarli definitivamente al calcolatore per la loro elaborazione.

Può essere interessante, a questo proposito, illustrarvi una tecnica molto affidabile per l'ingresso di grandissimi quantitativi di dati che prescinde completamente dagli errori che un operatore può commettere. Questa tecnica consiste nel lasciare completamente libero l'operatore di battere, alla sua massima velocità, i dati senza perdere tempo a correggere gli errori. Separatamente e indipendentemente un altro o-

peratore carica gli stessi dati. Il calcolatore confronta poi i due insiemi di dati e scarta i valori che non sono eguali. Sorprendentemente si verifica, in pratica, che le discordanze sono pochissime; la probabilità che i due operatori commettano poi gli stessi errori è molto bassa! Successivamente un secondo programma si occupa di richiedere l'ingresso di quei dati che sono risultati discordi.

Ingresso dati interattivo

Un programma per l'ingresso dei dati diviene interattivo quando fornisce all'operatore delle istruzioni e dei messaggi di pronto. Per darvi un esempio, modifichiamo un programma che abbiamo già usato in questo capitolo:

```
200 C$ = "W"
210  FOR I = 1 TO 800
220  PRINT C$;
230  NEXT I
240  PRINT "UFFA!"
250  END
```

Come abbiamo già visto questo programma visualizza 800 caratteri W seguiti dall'esclamazione UFFA! Esso funziona egualmente in Integer BASIC e in Applesoft.

Supponiamo di volere visualizzare un carattere qualunque invece della W. Per prima cosa cancelliamo la linea 200 per togliere l'assegnazione di W (battete 200 e subito il ritorno carrello). Otteniamo così:

```
210  FOR I = 1 TO 800
220  PRINT C$;
230  NEXT I
240  PRINT "UFFA!"
250  END
```

Battiamo ora in modo immediato C$ = "X" e poniamo in esecuzione il programma:

```
]C$="X"

]RUN
UFFA!
```

Come vedete, sullo schermo appare solamente la parola UFFA! e nessuna X. Ciò è dovuto al fatto che il valore "X" non è stato *trasmesso* al programma a causa del comando RUN che azzera tutte le variabili numeriche e annulla tutte le stringhe. Di conseguenza C$ è stata annullata e sullo schermo sono apparsi 800 caratteri nulli.

Per far giungere il nuovo valore di C$ al programma è possibile usare GOTO invece

di RUN. Infatti se imponiamo il salto GOTO 210, dopo aver dato un valore a C$, il programma funziona correttamente:

```
]C$="X"

]GOTO 210
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
UFFA!
```

Un altro modo, senz'altro più corretto, per dare il valore corrente a C$ è quello di inserire una istruzione INPUT diretttamente nel programma. Battete infatti:

```
]200 INPUT C$
```

Provate ora a listare il programma:

```
200  INPUT C$
210  FOR I = 1 TO 800
220  PRINT C$;
230  NEXT I
240  PRINT "UFFA!"
250  END
```

Ponetelo quindi in esecuzione. Il cursore scende di una linea (in Applesoft appare anche un punto interrogativo). Battete un solo carattere e premete RETURN. Il carattere che avete scelto viene visualizzato 800 volte. Rieseguite il programma con un altro carattere ancora. Notate che se per errore battete due caratteri, in Integer BA-

SIC, vi appare l'avviso di errore * * * STR OVFL ERR. Ciò è dovuto al fatto che in Integer BASIC dovete sempre dimensionare le stringhe con più di un carattere.

Sebbene il nostro programma sia adesso molto più professionale, è possibile migliorarlo ancora inserendovi un avviso per l'operatore. Battete allora questa nuova linea:

```
]190 PRINT "FORNIRE UN CARATTERE"
```

Con LIST provate a controllare se non avete commesso errori.

Il programma è ora corretto, ma se vogliamo eseguirlo più volte dobbiamo sempre dare il comando RUN. Si può ovviare a questo inserendo una istruzione di salto che riporti il programma al suo inizio:

```
]250 GOTO 190
```

Potrete adesso visualizzare tutti i caratteri che volete con la massima semplicità. Per terminare dovete battere però CTRL-C. Anche quest'ultimo inconveniente può essere eliminato mediante una istruzione IF-THEN che verifichi se l'operatore batte un carattere nullo per voler terminare il programma. Ecco le istruzioni necessarie:

```
190 PRINT "FORNIRE UN CARATTERE"
200 INPUT C$
205 IF C$ = "" THEN END
210 FOR I=1 TO 800
220 PRINT C$;
230 NEXT I
240 PRINT "UFFA!"
250 GOTO 190
```

Alla linea 205 viene controllato se C$ ha il valore nullo. Questo si ottiene battendo subito RETURN a seguito della INPUT della linea 200.

Prima di memorizzare questo programma è opportuno però inserire in esso tutte le istruzioni REM necessarie per spiegare il suo funzionamento. Vi ricordiamo che le REM possono essere poste su linee separate o su quelle già esistenti purchè separate dai due punti (:).

```
210 FOR I=1 TO 800: REM 800/40=20 LINEE
```

Il programma completo può essere come quello della Figura 4.3.

```
10   REM ******** PROVA ********
20   REM VISUALIZZAZIONE CONTINUA
30   REM DI UN CARATTERE BATTUTO
40   REM ALLA TASTIERA
50   REM **********************
190  PRINT "BATTERE UN CARATTERE"
200  INPUT C$
203  REM IL PROGRAMMA TERMINA CON L'INGRESSO NULLO
205  IF C$ = "" THEN  END
210  FOR I = 1 TO 800: REM 800/40=20 LINEE
220  PRINT C$;
230  NEXT I
240  PRINT "UFFA!"
250  GOTO 190
```

Figura 4.3 — Programma PROVA.

Messaggi di avviso

Ogni programma che richieda dei dati in ingresso deve mandare dapprima dei messaggi all'operatore. Spesso a questi messaggi basta rispondere con un SI o un NO (in inglese YES o NO). Un esempio di messaggio può essere questo:

VUOI FARE UN CAMBIAMENTO?

L'operatore risponderà con un YES (SI) o con un NO.
In un altro caso all'operatore viene chiesto di fare una scelta fra più casi possibili:

QUALE DATO DESIDERI CAMBIARE?

I programmi che sovrintendono a questo tipo di dialogo sono normalmente scritti come subroutine. Di conseguenza possono verificarsi tre diverse situazioni che necessitano di una particolare attenzione:

1. Non è prevedibile dove il messaggio di avviso apparirà sullo schermo, per cui esso può sovrapporsi ad un testo precedente. Avremo quindi una riga dello schermo su cui sarà presente il nostro messaggio più eventuali caratteri vecchi. Proprio questi vecchi caratteri possono portare a errori imprevedibili. È necessario quindi annullare in antiticpo la zona dello schermo che ci interessa. Ecco per esempio le istruzioni per annullare ER caratteri:

```
5000 REM ELIMINA GLI SPAZI
5010 FOR I=1 TO ER: PRINT "";:NEXT I:RETURN
```

2. Siccome usiamo delle subroutine, vi saranno sicuramente dei parametri che dovranno essere trasferiti tra la subroutine stessa e il programma principale. Per esempio il programma principale può avvisare la subroutine di accettare in ingresso solo dati entro un certo intervallo.

3. La subroutine deve sempre ritornare al programma principale la risposta dell'operatore.

La logica di una subroutine non può prevedere dove apparirà sullo schermo il messaggio di avviso. È importante quindi che il programma principale posizioni il cursore nel punto dello schermo più opportuno prima di chiamare la subroutine.

Vediamo ora un esempio. Supponiamo che una subroutine ponga all'operatore una domanda a cui si dovrà rispondere con un Yes o un No. La domanda verrà posta mediante una PRINT e la risposta sarà ricevuta con una INPUT. La zona dello schermo interessata a questo dialogo verrà opportunamente cancellata in precedenza. Ecco il programma:

```
80 N
100  REM  MUOVE IL CURSORE
140  VTAB 1
150  REM   ** CHIEDE RISPOSTA Y/N **
160  GOSUB 3020
170  END
3000  REM   ** RISPOSTA Y/N **
3020 C =  POS (0): REM COLONNA CURSORE
3030 R =  PEEK (37): REM  RIGA CURSORE
3040  REM  PREPARA LO SPAZIO
3050 ER = 35
3060  GOSUB 5010
3070  HTAB C + 1: REM RIPOSIZ. CURSORE
3080  PRINT "VUOI FARE UN CAMBIAMENTO?"
3090  INPUT R$
3100  IF R$ = "Y" OR R$ = "N" THEN  RETURN
3110  REM  RISPOSTA ERRATA
3120  VTAB R + 1: REM RIPOSIZ.CURSORE VERT.
3130  GOTO 3050
5000  REM  ** CANCELLA SPAZI **
5010  FOR I = 1 TO ER: PRINT " ";: NEXT I: RETURN
```

Analizziamo adesso il caso in cui l'operatore deve rispondere con un numero. La subroutine verificherà che la risposta sia superiore ad un valore minimo LO ed inferiore ad uno massimo HI. Il valore accettato viene ritornato tramite la variabile NM.

Questo è il programma:

```
100  REM  INTERVALLO INGRESSO
140 LO = 1:HI = 10
150  VTAB 1
170  GOSUB 3500
180  END
3500  REM ** RICHIESTA DATO NUMERICO **
3510  REM **  LO <= DATO <= HI **
3520  REM ** NM E' IL DATO NUMERICO **
3530  GOSUB 5110
3540  REM PULISCE SCHERMO PER INGRESSO
3550 ER = 35
3560  GOSUB 5010
3570  HTAB C + 1: REM RIPOSIZ. CURSORE
3580  PRINT "QUALE CAMPO VUOI CAMBIARE (1-10)?"
3590  INPUT NM
3600  IF NM >  = LO AND NM <  = HI THEN  RETURN
3610  REM RISPOSTA ERRATA
3620  VTAB R + 1: REM RIPOSIZ. CURSORE
3630  GOTO 3550: REM  RIPROVA
5000  REM ** PULISCE SPAZI INGRESSO **
5010  FOR I = 1 TO ER: PRINT " ";: NEXT I: RETURN
5100  REM ** RICORDA POSIZ. CURSORE **
5110 C =  PEEK (36): REM COLONNA
5120 R =  PEEK (37): REM  RIGA
5130  RETURN
```

In questo esempio il valore numerico dato in risposta deve avere una cifra sola. Provate per esercizio a modificare il programma in modo che tale valore possa avere due cifre. Se non riuscite a scrivere queste modifiche, attendete di vedere l'esempio, in cui si gestisce l'ingresso di una data, che vi illustreremo più avanti in questo capitolo.

Un altro miglioramento può essere il seguente. Il messaggio di avviso, che fa stampare la subroutine, è contenuto come stringa in una istruzione della subroutine stessa. Provate a fare in modo che questa stringa sia trasferita alla subroutine dal programma principale così da rendere l'uso della subroutine più generale.

Controllo degli errori

Se volete scrivere programmi veramente professionali, dovete fare del vostro meglio per evitare che un errore dell'operatore ne rovini irreparabilmente la sua esecuzione.

In alcuni casi è lo stesso calcolatore Apple II che rifiuta dati errati in ingresso come per esempio se date valori alfabetici, al posto di quelli numerici, a seguito di una INPUT A. In questo caso infatti vi verrà segnalato un messaggio di errore e sarete invitati a ribattere il valore in ingresso.

Purtroppo però le possibilità di controllo, insite nel calcolatore, sono molto poche per cui è necessario che siate voi a programmarle. Ecco un esempio che chiarisce il problema:

```
100 INPUT X
200 PRINT 100/X
300 END
```

Se date il valore zero in ingresso commetterete ovviamente l'errore di fare una divisione per zero. Per evitare questo è sufficiente porre una istruzione di controllo IF-THEN per escludere il valore X=0:

```
110 IF X<>0 THEN 200
120 PRINT "NON CONSENTITO - RIPETERE"
130 GOTO 100
```

Generalizzando quanto abbiamo appena illustrato si capisce come è possibile controllare la validità di un dato in ingresso. Invece di usare le istruzioni IF-THEN si possono usare in Applesoft le istruzioni ON-GOTO oppure ON-GOSUB e in Integer BASIC le istruzioni GOTO e GOSUB calcolate. Nel prossimo paragrafo faremo degli esempi molto più dettagliati su questo argomento.

Le istruzioni ONERR GOTO e RESUME

In Applesoft esiste una speciale istruzione che permette di fare dei salti condizionati qualora si verifichi un errore. Ecco un esempio:

```
50 ONERR GOTO 8000
```

Questa istruzione permette di saltare alla linea indicata, di visualizzare un messaggio per l'operatore e di fermare l'esecuzione del programma. Essa pone anche un numero in codice, corrispondente al tipo di errore, nella posizione di memoria 222. Con una successiva istruzione PEEK è possibile allora andare a leggere questo codice. Nella Tabella C.1 dell'Appendice C sono riportate tutte le condizioni di errore che si possono rilevare con l'istruzione ONERR GOTO.

L'uso più opportuno che potete fare dell'istruzione ONERR GOTO è quello di scrivere una routine che gestisca la situazione di errore e cerchi di prendere i necessari provvedimenti. Al termine di questa routine potete porre una istruzione RESUME che rimanda al punto dove si era verificato l'errore. Altrimenti potete porre una normale GOTO per trasferire l'esecuzione del programma a qualunque numero di linea. La

routine di correzione dipende quindi dal tipo di errore che volete tenere sotto controllo e dallo stato del programma in quel momento. Lo stato del programma viene normalmente controllato ispezionando alcune variabili più importanti.

Per annullare il salto condizionato ONERR GOTO, e ripristinare la gestione automatica degli errori da parte dell'Apple II, dovete dare il comando POKE 216,0.

Nell'esempio seguente viene impiegata l'istruzione ONERR GOTO per controllare l'ingresso dei dati. Se l'errore riguarda l'ingresso dei dati vengono visualizzati dei messaggi di avviso e viene ripetuta la richiesta dei dati, se invece l'errore è di altro tipo il programma viene terminato.

```
50  ONERR  GOTO 8000
200  PRINT "DARE UN VALORE DI STRINGA"
210  INPUT X$
220  PRINT "DARE UN VALORE NUMERICO"
230  INPUT X
240  PRINT "DARE UN VALORE INTERO"
250  INPUT X%
260  GOTO 200
500  REM FINE DEL PROGRAMMA
510  PRINT "ULTIMI VALORI DATI SONO ";X$;", ";X;" E ";X%
515  POKE 216,0: REM  DISABILITA ONERR
520  END
8000  REM ** ROUTINE GESTIONE ERRORE **
8010 E =  PEEK (222): REM  NUMERO ERRORE
8020  IF E = 255 THEN  GOTO 500: REM  TERMINA CON CTRL-C
8030  IF E = 53 OR E = 176 OR E = 254 THEN 8100
8040  INVERSE
8050  PRINT "ERRORE N. ";E;" !"
8055  PRINT "PRENDI NOTA DI QUESTO NUMERO"
8060  PRINT "E SENZA SPEGNERE IL CALCOLATORE"
8070  PRINT "CERCA DI CAPIRNE LA CAUSA."
8090  NORMAL : STOP
8100  REM  ERRORE RICONOSCIUTO
8110  PRINT "": REM  "CTRL-G"
8130  PRINT "ERRORE...RIPROVA"
8140  RESUME
```

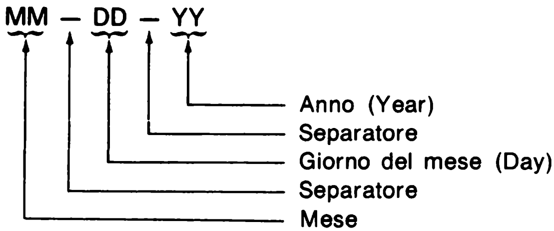
Ingresso e verifica di una data

Sviluppiamo ora un programma che farà uso di molte delle tecniche che abbiamo appena descritto. Per brevità usiamo il BASIC dell'Applesoft, ma voi stessi potete convertirlo in Integer BASIC.

Molti programmi durante la loro esecuzione richiedono all'operatore dei dati che non sono così semplici come un "sì" o un "no", ma neanche tanto complicati come un intero testo. Consideriamo per esempio la data.

Non sottovalutate questo esempio perchè contiene tutti gli elementi caratteristici di una buona procedura per l'ingresso dei dati.

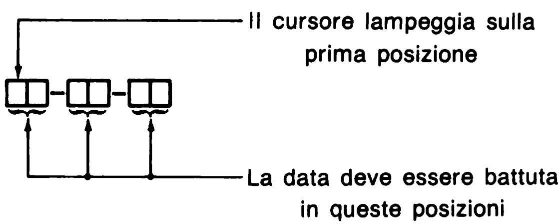
Poniamo allora che la data debba essere fornita così (con il mese prima del giorno come è d'uso nei paesi anglosassoni):



I tre valori MM, DD e YY sono forniti come numeri con due cifre senza il RETURN terminale.

Se volete potete sostituire il trattino con una barra o un punto.

Programmate l'ingresso dei dati sempre in modo che esso sia facile e immediato per l'operatore! Per esempio nel nostro caso predisponete i campi dove devono essere battuti il mese, il giorno e l'anno mediante caratteri invertiti. Ecco come fare:



Questi tre campi li potete creare con il seguente programma:

```
5  REM  POSIZIONA PER L'INGRESSO
10  HOME : VTAB 3: HTAB 20
20 IW = 2: GOSUB 1100: REM INGRESSO 2 CARAT.
30  PRINT "-";
40  GOSUB 1100: REM INGRESSO 2 CARAT.
50  PRINT "-";
60  GOSUB 1100: REM INGRESSO 2 CARAT.
70  VTAB 3: HTAB 20
80  END
1090  REM VISUALIZZA "IW" SPAZI INVERSI
1100  INVERSE
1110  FOR I = 1 TO IW: PRINT " ";: NEXT I
1120  NORMAL
1130  RETURN
```

L'ingresso della data inizia alla colonna 21 della riga 3 dello schermo e la zona circostante viene pulita per evitare confusione con l'ingresso della data stessa. Dopo aver fatto apparire il campo d'ingresso della data il cursore torna all'inizio per l'ingresso dei dati veri e propri.

Provate a porre una INPUT alla linea 80 per ricevere il mese:

```
80 INPUT M$
90 END
```

Vi accorgerete subito però che appena premete RETURN per, confermare il mese, viene cancellato il resto della linea. In questa situazione è infatti preferibile usare l'istruzione GET. Aggiungete infatti queste linee:

```
80 GOSUB 1200:MM$=C$: REM 1A CIFRA MESE
90 GOSUB 1200:MM$=MM$+C$: REM 2A CIFRA MESE
1190 REM ** INGRESSO UN CARATTERE **
1200 GET C$
1210 PRINT C$;: REM ECO DEL CARATTERE
1220 RETURN
```

Queste istruzioni accettano due cifre in ingresso. L'ingresso è visualizzato nel primo campo invertito della data. Non è necessario battere RETURN in quanto il programma accetta solo i primi due caratteri che vengono battuti.

Per dare la data completa è necessario dare tre volte una coppia di numeri: una per il mese, una per il giorno e una per l'anno. Invece di ripetere tre volte queste stesse istruzioni le poniamo in una subroutine che richiamiamo tre volte.

```
80 GOSUB 1300:MM$=CC$: REM MESE
90 PRINT "-";: REM SPAZIATURA
180 GOSUB 1300:DD$=CC$: REM GIORNO
190 PRINT "-";: REM SPAZIATURA
280 GOSUB 1300:YY$=CC$: REM ANNO
290 END
1290 REM ** INGRESSO DUE CARATTERI **
1300 GOSUB 1200:CC$=CC$+C$: REM INGRESSO SECONDO CAR.
1320 RETURN
```

Le variabili MM$, DD$ e YY$ contengono il mese, il giorno e l'anno espressi come stringhe di due caratteri.

Ci sono due modi per aiutare l'operatore a correggere gli errori mentre sta caricando una data.

1. Il programma può controllare automaticamente la validità del mese, del giorno e dell'anno.
2. L'operatore può ripetere l'ingresso dei dati.

Il programma può controllare che il numero del mese sia compreso tra 1 e 12. Può controllare il numero massimo di giorni per ogni mese ed eventualmente tenere conto degli anni bisestili. Ogni anno tra 00 e 99 è valido. Se una di queste condizioni non è rispettata allora l'ingresso della data dovrà essere ripetuta dall'inizio.

Se l'operatore preme RETURN, tutta la procedura di ingresso della data viene ripetuta.

In definitiva il programma è il seguente:

```
5  REM  POSIZIONA PER L'INGRESSO
10  HOME : VTAB 3: HTAB 20
15 RC$ =  CHR$ (13): REM  CARATTERE DI RIPARTENZA
20 IW = 2: GOSUB 1100
30  PRINT "-";
40  GOSUB 1100
50  PRINT "-";
60  GOSUB 1100
65  PRINT "";: REM "CTRL-G"
68  REM TRE CAMPI D'INGRESSO
70  VTAB 3: HTAB 20
80  REM MESE
90  GOSUB 1300: IF C$ = RC$ THEN  GOTO 10: REM  CONTROLLO
    RIPARTENZA
100 MM =  VAL (CC$): REM  NUMERO MESE
110  IF MM < 1 OR MM > 12 THEN 10
120 DT$ = CC$: PRINT "-";
125  REM NUMERO MAX GIORNI/MESE
130 DM = 31: REM MESI DI 31 GIORNI
140  IF MM = 2 THEN DM = 29: REM  FEBBRAIO
150  REM  MESI DI 30 GIORNI
160  IF MM = 4 OR MM = 6 OR MM = 9 OR MM = 11 THEN DM = 30
170  REM GIORNO
180  GOSUB 1300: IF C$ = RC$ THEN  GOTO 10
190 DD =  VAL (CC$): IF DD < 1 OR DD > DM THEN 10
200 DT$ = DT$ + "-" + CC$: PRINT "-";
270  REM  ANNO
280  GOSUB 1300: IF C$ = RC$ THEN  GOTO 10
290 YY =  VAL (CC$): IF YY < 0 OR YY > 99 THEN 10
300 DT$ = DT$ + "-" + CC$
390  REM VISUALIZZAZIONE
400  VTAB 10: HTAB 19: PRINT "LA DATA E'"
410  VTAB 11: HTAB 20: PRINT DT$
420  END
1090  REM VISUALIZZA "IW" SPAZI INVERSI
1100  INVERSE
1110  FOR I = 1 TO IW: PRINT " ";: NEXT I
1120  NORMAL
1130  RETURN
1190  REM INGRESSO UN CARATTERE
1200  GET C$: IF C$ = "" THEN 1200
1210  IF C$ = RC$ THEN  RETURN
1220  IF C$ < "0" OR C$ > "9" THEN  GOTO 1200
```

```
1230  PRINT C$;: REM  ECO DEL CARATTERE
1240  RETURN
1290  REM INGRESSO DUE CARATTERI
1300  REM INGRESSO PRIMO CARATTERE
1310  GOSUB 1200: IF C$ = RC$ THEN  RETURN
1315 CC$ = C$
1320  REM INGRESSO SECONDO CARATTERE
1330  GOSUB 1200: IF C$ = RC$ THEN  RETURN
1335 CC$ = CC$ + C$
1340  RETURN
```

Notate che la data è posta nella stringa di otto caratteri DT$ come mese, giorno e anno.

Sulla data fornita in ingresso vengono fatti questi tre controlli:

1. Il carattere è RETURN?
2. Se non è RETURN, è una cifra valida?
3. La prima coppia di cifre costituisce un mese valido, la seconda un giorno valido e la terza un anno valido?

Abbiamo scelto RETURN come carattere di ripristino. Cambiando CHR$ (13), alla linea 15, potete porre un qualunque altro carattere di ripristino. Quando questo carattere viene battuto tutta la procedura di ingresso della data viene ripetuta. Dobbiamo controllare il carattere di ripristino sia nella subroutine d'ingresso della linea 1200, sia in quella della linea 1300 perchè desideriamo fare ripartire il programma dopo l'ingresso sia della prima che della seconda cifra. Anche il programma principale controlla il carattere di ripristino ed eventualmente ritorna alla linea 10. Potreste saltare direttamente, dalla subroutine della linea 1200, alla linea 10, ma questo significherebbe usare una GOTO invece della istruzione RETURN.

Ogni subroutine deve essere vista come una unità autonoma con i suoi punti fissi di ingresso e di uscita. Saltare da un punto qualunque di una subroutine al programma principale, o da una subroutine ad un'altra, è sicuramente motivo di gravi errori. In altre parole per uscire da una subroutine dovete sempre passare attraverso la sua istruzione finale RETURN. (Ricordate che se siete usciti direttamente da una subroutine dovete cancellare l'indirizzo di ritorno con l'istruzione POP).

La routine per il controllo di dati non numerici, forniti in ingresso, è posta alla linea 1220.

Il controllo di validità dei mesi, dei giorni e degli anni viene fatto nel programma principale (non nella subroutine) poichè ogni coppia di cifre ha estremi di validità diversi.

l'istruzione alla linea 110 controlla la validità del mese.

Le istruzioni 130, 140 e 160 calcolano il numero massimo di giorni per ogni mese, mentre l'istruzione 190 controlla la validità del giorno.

Alla linea 290 viene fatto il controllo dell'anno.

Ricordatevi infine che se cercate di scrivere bene una routine per l'ingresso dei dati, forse spendete un poco di tempo in più, ma ne farete risparmiare moltissimo a chi opererà con il vostro programma.

MODULI PER L'INGRESSO DEI DATI

In questo paragrafo esamineremo alcune tecniche di programmazione che si possono usare molto bene con l'Applesoft. Molte delle cose che ora vedremo non si possono usare però in Integer BASIC. Se lavorate solo con l'Integer BASIC leggete egualmente questo paragrafo per ampiare la vostra esperienza di programmazione e tentare, eventualmente, di tradurre in questo tipo di BASIC le tecniche che vi insegnamo.

Se avete dei dati d'ingresso complicati, il modo migliore per caricarli è quello di visualizzare sullo schermo un modulo e quindi riempirlo mano mano con i dati. Consideriamo per semplicità di dover caricare nomi e indirizzi. Potreste allora visualizzare un modulo come questo

```
FORNITE NOME E INDIRIZZO
          1  NOME:
          2  VIA:
          3  CITTÀ:
          4  STATO:
          5  CAP:
```

Ad ogni campo d'ingresso viene assegnato un numero che appare sullo schermo con caratteri invertiti. L'operatore deve porre con ordine i dati richiesti.

Il programma seguente pulisce lo schermo e visualizza il modulo:

```
109  REM
110  REM DISPLAY THE DATA ENTRY FORM
111  REM
120  CALL  - 936: VTAB 2: REM  CLEAR SCREEN AND POSITION
 CURSOR
125  PRINT "ENTER NAME AND ADDRESS BELOW"
130  REM FIRST DISPLAY FIELD NUMBERS
140  INVERSE
150  FOR I = 1 TO 4: HTAB 2: PRINT I: NEXT I
160  VTAB 6: HTAB 29: PRINT 5
170  NORMAL
180  REM NOW DISPLAY FIELD NAMES
190  VTAB 3: HTAB 6: PRINT "NAME:"
200  HTAB 4: PRINT "STREET:"
210  HTAB 6: PRINT "CITY:"
220  HTAB 5: PRINT "STATE:";
230  HTAB 31: PRINT "ZIP:"
```

Appena un dato viene battuto, viene creato un campo invertito per indicare dove apparirà il dato stesso. Mediante CTRL-X è possibile ripetere il dato del campo in quel momento in uso. Il tasto RETURN pone termine all'ingresso in ogni singolo campo. Ecco il programma per ottenere quanto descritto:

```
100 RC$ =  CHR$ (24): REM  CTRL-X IS THE RESTART CHAR.
299  REM
300  REM ENTER ALL 5 FIELDS
301  REM
310  FOR F = 1 TO 5: GOSUB 1900: NEXT F
320  END
990  REM +++++++++++SUBROUTINE 1000+++++++++++
991  REM  ENTER STRING DATA INTO A FIELD WITH LN CHARACTERS
992  REM THE CURSOR MUST BE IN THE FIELD'S FIRST POSITION
993  REM THE RETURN KEY WILL END DATA ENTRY
994  REM THE 'LEFT ARROW' KEY RESTARTS ENTRY
995  REM NO VALIDITY CHECKS ON ENTERED DATA
996  REM THE ENTERED STRING IS RETURNED IN CC$
997  REM
1000 HT =  POS (0) + 1: REM   REMEMBER START-OF-FIELD
 POSITION
1010  REM  DISPLAY INVERSE VIDEO ENTRY MASK
1020  INVERSE
1030  FOR I = 1 TO LN: PRINT " ";: NEXT I
1040  NORMAL : HTAB (HT): REM  REPOSITION TO START OF FIELD
1050  REM  ENTER DATA
1060 CC$ = "": REM  INITIALIZE OUTPUT TO NULL
1070  GET C$
1080  IF C$ = RC$ THEN  HTAB (HT): GOTO 1020: REM  RESTART
 ENTRY?
1090  IF C$ =  CHR$ (13) THEN  GOTO 1140: REM  END OF ENTRY?
1100  REM   WHEN ENTRY IS FULL, WAIT FOR RETURN OR RESTART
1110  IF  LEN (CC$) = LN THEN  GOTO 1070
1120  PRINT C$;: REM  ECHO KEYSTROKE
1130 CC$ = CC$ + C$: GOTO 1070
1135  REM   ENTRY FINISHED, FILL THE REST OF CC$ WITH BLANKS
1140 J =  LEN (CC$)
1150  FOR I = J TO LN:CC$ = CC$ + " ": NEXT I
1160  REM REDISPLAY ENTRY
1170  HTAB (HT): PRINT CC$;: RETURN
1889  REM +++++++++++SUBROUTINE 1900+++++++++++
1890  REM   BRANCH TO ENTRY ROUTINE FOR FIELD NUMBER F
1891  REM
1900  ON F GOTO 2000,2100,2200,2300,2400: RETURN
1989  REM
1990  REM ENTER 20-CHAR. NAME
1991  REM
2000  VTAB 3: HTAB 11
2010 LN = 20: GOSUB 1000:NA$ = CC$: RETURN
2089  REM
2090  REM  ENTER 20-CHAR STREET
2091  REM
```

```
2100  VTAB 4: HTAB 11
2110 LN = 20: GOSUB 1000:CI$ = CC$: RETURN
2189  REM
2190  REM  ENTER 20-CHAR. CITY
2191  REM
2200  VTAB 5: HTAB 11
2210 LN = 20: GOSUB 1000: RETURN
2289  REM
2290  REM  ENTER 18-CHAR. STATE
2291  REM
2300  VTAB 6: HTAB 11
2310 LN = 18: GOSUB 1000:ST$ = CC$: RETURN
2389  REM
2390  REM  ENTER 5-CHAR. ZIP CODE
2391  REM
```

Caricate quindi questo programma dalla linea 100 alla 2391. Se avete caricato in precedenza le istruzioni dalla linea 109 alla 230 non è necessario ribatterle adesso.

Ponetelo quindi in esecuzione e se qualcosa non funziona controllate di avere copiato esattamente tutte le istruzioni; fate attenzione particolarmente ai punti e virgola delle PRINT!

Quando lo ponete in esecuzione ognuno dei cinque campi d'ingresso sarà reso più luminoso dall'inversione del campo. Alla fine, premendo RETURN, il campo dei caratteri diverrà normale. Provate anche a premere CTRL-X per ricominciare l'ingresso dei dati.

Prima di procedere oltre esaminate attentamente la subroutine per l'ingresso dei dati, che inizia alla linea 1000 e termina alla 1170, e cercate di capirne bene la struttura logica. Notate quanto sia facile per l'operatore capire quali dati deve dare in ingresso ed eventualmente correggerli.

Dopo che un intero nome ed indirizzo è stato battuto, il programma chiederà all'operatore se desidera fare qualche cambiamento e in tal caso quale campo deve essere modificato. Le subroutine per porre queste domande tipo sì/no le abbiamo già descritte all'inizio di questo capitolo. In questo caso ne usiamo però una versione modificata in cui il programma chiamante fornisce le domande di chiedere all'operatore. Ecco il listato completo del programma:

```
9  REM ***************************************************
10  REM THIS PROGRAM DISPLAYS A FORM FOR ENTERING A NAME AND
11  REM ADDRESS THEN IT REQUESTS ENTRY OF THAT DATA
12  REM **************************************************
13  REM
100 RC$ =  CHR$ (24): REM  CTRL-X IS THE RESTART CHAR.
109  REM
110  REM DISPLAY THE DATA ENTRY FORM
111  REM
120  CALL  - 936: VTAB 2: REM  CLEAR SCREEN AND POSITION
 CURSOR
```

```
125  PRINT "ENTER NAME AND ADDRESS BELOW"
130  REM FIRST DISPLAY FIELD NUMBERS
140  INVERSE
150  FOR I = 1 TO 4: HTAB 2: PRINT I: NEXT I
160  VTAB 6: HTAB 29: PRINT 5
170  NORMAL
180  REM NOW DISPLAY FIELD NAMES
190  VTAB 3: HTAB 6: PRINT "NAME:"
200  HTAB 4: PRINT "STREET:"
210  HTAB 6: PRINT "CITY:"
220  HTAB 5: PRINT "STATE:";
230  HTAB 31: PRINT "ZIP:"
299  REM
300  REM ENTER ALL 5 FIELDS
301  REM
310  FOR F = 1 TO 5: GOSUB 1900: NEXT F
319  REM
320  REM  ALLOW CHANGES
321  REM
330  VTAB 23: HTAB 1: REM  GET ENTRY ON BOTTOM LINE
340 QU$ = "DO YOU WANT TO MAKE ANY CHANGES? "
350  GOSUB 1300: REM GET Y/N RESPONSE
360  IF YN$ = "N" THEN  GOTO 500
370  VTAB 23: HTAB 1: REM GET ENTRY ON BOTTOM LINE
380 QU$ = "ENTER NUMBER OF FIELD TO CHANGE "
390 LO = 1:HI = 5
400  GOSUB 1400: REM GET NUMERIC RESPONSE
410 F = NM: GOSUB 1900: REM CHANGE FIELD F
420  GOTO 330
490  REM  END OF PROGRAM
500  VTAB 23: HTAB 1: GOSUB 1200: REM CLEAR BOTTOM LINE
510  END
990  REM ++++++++++SUBROUTINE 1000+++++++++++
991  REM  ENTER STRING DATA INTO A FIELD WITH LN CHARACTERS
992  REM THE CURSOR MUST BE IN THE FIELD'S FIRST POSITION
993  REM THE RETURN KEY WILL END DATA ENTRY
994  REM THE 'LEFT ARROW' KEY RESTARTS ENTRY
995  REM NO VALIDITY CHECKS ON ENTERED DATA
996  REM THE ENTERED STRING IS RETURNED IN CC$
997  REM
1000 HT =  POS (0) + 1: REM  REMEMBER START-OF-FIELD POSITION
1010  REM  DISPLAY INVERSE VIDEO ENTRY MASK
1020  INVERSE
1030  FOR I = 1 TO LN: PRINT " ";: NEXT I
1040  NORMAL : HTAB (HT): REM  REPOSITION TO START OF FIELD
1050  REM  ENTER DATA
1060 CC$ = "": REM  INITIALIZE OUTPUT TO NULL
1070  GET C$
1080  IF C$ = RC$ THEN  HTAB (HT): GOTO 1020: REM  RESTART
 ENTRY?
1090  IF C$ =  CHR$ (13) THEN  GOTO 1140: REM  END OF ENTRY?
1100  REM   WHEN ENTRY IS FULL, WAIT FOR RETURN OR RESTART
1110  IF  LEN (CC$) = LN THEN  GOTO 1070
```

```
1120  PRINT C$;: REM  ECHO KEYSTROKE
1130 CC$ = CC$ + C$: GOTO 1070
1135  REM   ENTRY FINISHED, FILL THE REST OF CC$ WITH BLANKS
1140 J =  LEN (CC$)
1150  FOR I = J TO LN:CC$ = CC$ + " ": NEXT I
1160  REM REDISPLAY ENTRY
1170  HTAB (HT): PRINT CC$;: RETURN
1189  REM +++++++++++SUBROUTINE 1200+++++++++++
1190  REM  CLEAR ROW WHICH THE CURSOR IS ON
1191  REM
1200  HTAB 1: REM START AT BEGINNING OF ROW
1210  FOR I = 1 TO 39: PRINT " ";: NEXT I
1220  HTAB 1: REM LEAVE CURSOR AT BEGINNING OF ROW
1230  RETURN
1289  REM +++++++++++SUBROUTINE 1300+++++++++++
1290  REM ASK A QUESTION (QU$) AND RETURN A Y OR N RESPONSE
 IN YN$
1291  REM
1300  GOSUB 1200: REM  CLEAR ENTRY LINE
1310  PRINT QU$;: REM  DISPLAY PROMPT
1320  GET YN$: IF YN$ <  > "N" AND YN$ <  > "Y" THEN  GOTO
 1320
1330  PRINT YN$;: REM ECHO RESPONSE
1340  RETURN
1389  REM +++++++++++SUBROUTINE 1400+++++++++++
1390  REM ASK FOR NUMERIC ENTRY (PROMPT IS QU$)
1391  REM RETURN RESPONSE IN NM
1392  REM NM MUST BE <=HI AND >=LO
1393  REM
1400  GOSUB 1200: REM  CLEAR ENTRY LINE
1410  PRINT QU$;: REM DISPLAY PROMPT
1420  GET C$:NM =  VAL (C$)
1425  REM  CHECK THAT ENTRY IS WITHIN RANGE
1430  IF NM < LO OR NM > HI THEN  GOTO 1420
1440  PRINT C$;: REM  ECHO RESPONSE
1450  RETURN
1889  REM +++++++++++SUBROUTINE 1900+++++++++++
1890  REM   BRANCH TO ENTRY ROUTINE FOR FIELD NUMBER F
1891  REM
1900  ON F GOTO 2000,2100,2200,2300,2400: RETURN
1989  REM
1990  REM ENTER 20-CHAR. NAME
1991  REM
2000  VTAB 3: HTAB 11
2010 LN = 20: GOSUB 1000:NA$ = CC$: RETURN
2089  REM
2090  REM  ENTER 20-CHAR STREET
2091  REM
2100  VTAB 4: HTAB 11
2110 LN = 20: GOSUB 1000:STT$ = CC$: RETURN
2189  REM
2190  REM  ENTER 20-CHAR. CITY
2191  REM
```

```
2200  VTAB 5: HTAB 11
2210 LN = 20: GOSUB 1000:CI$=CC$:RETURN
2289  REM
2290  REM   ENTER 17-CHAR. STATE
2291  REM
2300  VTAB 6: HTAB 11
2310 LN = 17: GOSUB 1000:ST$ = CC$: RETURN
2389  REM
2390  REM  ENTER 5-CHAR. ZIP CODE
2391  REM
2400  VTAB 6: HTAB 35
2410 LN = 5: GOSUB 1000:ZI$ = CC$: RETURN
```

Studiate attentamente questo programma cercando di capire la sua struttura logica. I punti di maggiore interesse sono:

1. Mediante la numerazione e l'inversione, dei campi d'ingresso, viene indicato chiaramente all'operatore quali e quanti dati deve fornire e dove porli.

2. Quando l'operatore effettua una correzione il campo inverso lo avvisa dove essa è stata effettuata.

3. L'operatore non è obbligato a battere gli spazi vuoti perchè questi sono posti automaticamente premendo RETURN.

4. In ogni momento l'operatore può ripetere la procedura d'ingresso con il comando CTRL-X.

5. Alle domande l'operatore deve rispondere solo con un carattere ben preciso: Y o N per "yes" ("sì") e "no" oppure con un numero tra 1 e 5 per selezionare il campo. Rispondere invece con un carattere qualunque è molto pericoloso perchè si rischia di dare una risposta errata se accidentalmente si tocca la tastiera.

Le opzioni che riportiamo qui di seguito, sono altri miglioramenti che avremmo potuto inserire nel nostro programma:

1. Controllare che il codice CAP (ZIP negli USA) sia solo numerico.

2. Chiedere una conferma all'operatore quando le risposte sono negative. Questo per dare un'altra possibilità di correzione se per caso non si avesse voluto dare la risposta NO.

3. Aggiungere un altro tasto che rigetti gli ultimi dati forniti e ripristini i precedenti. Infatti nel nostro programma se per caso viene corretto un campo esatto, l'operatore deve ricaricare il campo buono oltre che ripetere la correzione. Con questo nuovo tasto speciale dovrebbe essere possibile rigettare subito il dato errato e ripristinare quello precedente.

4. Abilitare il tasto ← per usarlo come tasto per tornare indietro di un carattere. Ogni volta che il tasto viene premuto, il cursore torna indietro di un posto

e l'ultimo carattere battuto viene sostituto sullo schermo da una maschera d'ingresso (in modo inverso); mentre nella stringa CC$, della subroutine d'ingresso, diviene uno spazio nullo. Ovviamente non può avvenire alcun spostamento all'indietro del cursore quando esso è già all'estremo sinistro del campo d'ingresso.

Provate, per esercizio, ad aggiungere voi stessi queste opzioni al programma per l'ingresso di nomi e indirizzi.

USCITA DI DATI CON FORMATO

Quando accendete il vostro calcolatore Apple II l'uscita dei dati va direttamente sullo schermo televisivo. Esistono però delle istruzioni che portano i dati in uscita su una stampante o su altre periferiche capaci di ricevere dati.

Anche se a prima vista può sembrare la stessa cosa, programmare uno schermo o una stampante presenta delle notevoli differenze. Pensate per esempio che una stampante ha normalmente un numero di colonne molto superiore a quello di uno schermo; che sullo schermo potete muovere il cursore come volete, ma non potete di certo muovere liberamente la testina di una stampante.

Stante queste differenze, la programmazione di uno schermo e di una stampante sono molto simili. Molte cose che diremo qui di seguito valgono, sia per lo schermo, come per una stampante. Se una cosa è valida solo per lo schermo televisivo lo diremo esplicitamente. Se volete lavorare anche con una stampante, vi consigliamo però di attendere di aver letto i paragrafi che riguardano espressamente le stampanti.

Programmare l'uscita dei dati è molto più semplice di quanto si deve fare invece per il loro ingresso. Il motivo essenziale è che non si deve tenere conto dell'interazione con l'operatore.

Ecco alcune regole generali:

1. Evitate di ammassare troppa informazione in un piccolo spazio.
2. Se i dati devono essere incolonnati, allineateli a destra o a sinistra, così che siano di facile lettura.
3. Usate i caratteri in modo inverso per porre in risalto titoli o punti di riferimento del testo. Evitate di usare il campo inverso per le stampanti.

Vi riportiamo qui di seguito alcuni consigli per evitare gli errori più comuni che si commettono quando si programmano dati in uscita:

1. Ricordatevi di usare i separatori nella lista di una PRINT; cioè il punto e virgola o la virgola.
2. Disegnate su un pezzo di carta la configurazione che deve assumere il testo di uscita prima di programmarlo. Nella Appendice L è riportato un disegno quadrettato che può servirvi per calcolare facilmente le righe e le colonne dello schermo. Evitate di fare lunghi tentativi direttamente sullo schermo per trovare il formato giusto per l'uscita dei vostri dati!

3. Se volete ripartire una variabile con indice in un certo numero di colonne fate attenzione che essa può non essere esattamente ripartibile. Per esempio supponete di avere la variabile N$ (I), cioè un *vettore*, che abbia 25 elementi e volete ripartirla su tre colonne. Potreste pensare di scrivere così:

```
100 FOR I=1 TO 25 STEP 3
200 REM COLONNA 1
 .
 .
 .
300 REM COLONNA 2
 .
 .
 .
400 REM COLONNA 3
 .
 .
 .
500 NEXT I
```

In tal caso commettereste l'errore di voler calcolare gli elementi 26 e 27 di N$ (I) che non esistono. Per ovviare a questo è sufficiente che controlliate la fine di N$ (I) con una istruzione IF-THEN nel modo seguente:

```
100 FOR I=LO TO HI STEP ST
 .
 .
 .
350 I=I+1
360 IF I>HI THEN 510
 .
 .
 .
500 NEXT
510 REM IL PROGRAMMA CONTINUA
```

Formazione di una finestra di dati sullo schermo

Quando si deve lavorare con un gran quantitativo di dati, è molto utile usare lo schermo come una finestra che si apre sui dati. Per operare in qusto modo i dati devono essere raggruppati in pagine ognuna delle quali possa essere portata sullo schermo. I programmi che usano questa tecnica devono avere delle routine separate per visualizzare gli indici di campo e i dati per ogni pagina. Questi programmi devono anche permettere all'operatore di passare da una pagina ad un'altra.

Molto spesso le variabili con indici sono usate per contenere moltissimi dati per cui lo schermo è usato come il mirino di una macchina fotografica per "puntare" solo ad una parte di questi dati. Pensate, per esempio, che i dati siano scritti su un grande tabellone e che non sia possibile vederli tutti assieme nel mirino della macchina fotografica. Dovrete allora muovere la macchina in alto e in basso, a destra e a sinistra per vederne un gruppo alla volta. Le finestre, sul nostro schermo, hanno appunto il compito di fare la stessa cosa con i nostri dati.

Illustriamo adesso un esempio in cui creiamo una finestra in Applesoft. Iniziamo con il definire una variabile con due indici (cioè una *matrice*) i cui elementi sono così definiti:

X% (I,J) = OIOJ

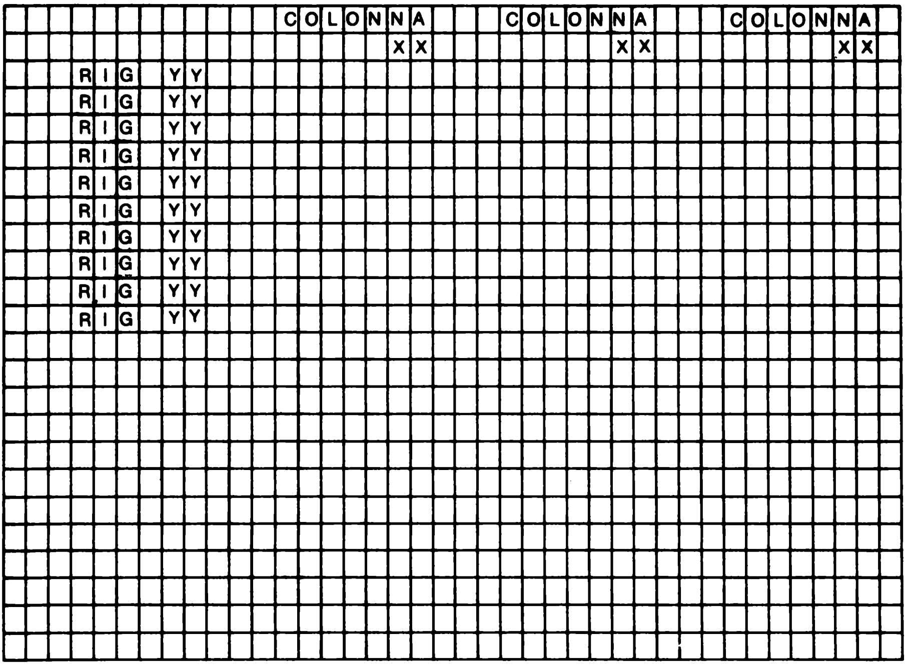
Per esempio:

X% (3,2) = 0302
X% (19,8) = 1908
X% (11,12) = 1112

Possiamo allora inizializzare questa matrice nel modo seguente:

```
10 DIM X%(14,50)
20 FOR I=1 TO 14
30 FOR J=1 TO 50
40 X%(I,J)=I*100 + J
50 NEXT J
60 NEXT I
```

Visualizziamo adesso una parte di questa matrice. Sulle prime due righe dello schermo metteremo le indicazioni delle colonne e sulle prime nove colonne di destra le indicazioni delle righe. Ecco un disegno illustrativo:



I numeri di colonna appariranno al posto delle XX e i numeri di riga al posto delle YY. Il programma seguente crea queste intestazioni in campo inverso:

```
1000  INVERSE
1020  FOR I = 1 TO 3
1030  HTAB 3 + I * 10: PRINT "COLONNA";
1040  NEXT I
1050  PRINT
1060  FOR I = 0 TO 2
1070 S% = 0: IF C% + I < 10 THEN S% = 1
1080  HTAB 18 + I * 10: PRINT  SPC( S%);C% + I;
1090  NEXT I
1100  PRINT
1110  FOR I = R% TO R% + 9
1120 S% = 0: IF I < 10 THEN S% = 1
1130  HTAB 4: PRINT "RIG";: HTAB 8: PRINT  SPC( S%);I
1140  NEXT I
1150  NORMAL : RETURN
```

Abbiamo deliberatamente creato una finestra più piccola di tutto lo schermo per meglio descrivere il concetto di finestra di dati. Se lo ritenete opportuno potete estenderla a tutto lo schermo. Tuttavia avrete spesso la necessità di aprire finestre più piccole del vostro schermo per mantenere disponibile una parte di esso per altri dati.

Aggiungiamo ora le istruzioni per chiedere all'operatore quali sono i valori minimi, di riga e di colonna, che vuole visualizzare. Tale elemento della matrice apparirà allora in alto a sinistra e la finestra sarà completata con gli elementi successivi. Ecco il programma completo:

```
10  HOME
15  PRINT "ATTENDERE INIZIALIZZAZIONE PROGRAMMA"
20  DIM X%(14,50)
30  FOR I = 1 TO 14
40  FOR J = 1 TO 50
50 X%(I,J) = I * 100 + J
60  NEXT J
70  NEXT I
75  HOME
80  HTAB 1: VTAB 20: INPUT "DARE LA COLONNA (1-12):";C%
90  IF C% < 1 OR C% > 12 THEN  GOTO 80
100  VTAB 21: INPUT "DARE LA RIGA (1-41):";R%
110  IF R% < 1 OR R% > 41 THEN  GOTO 100
120  VTAB 1: HTAB 1: GOSUB 1000
135  VTAB 3
140  FOR I = R% TO R% + 9
150  HTAB 10
160  FOR J = C% TO C% + 2
165  REM VISUALIZ. VALORI ALLINEATI A DESTRA
170 X$ =  STR$ (X%(J,I)): PRINT  SPC( 10 -  LEN (X$));X$;
180  NEXT J
190  PRINT
```

```
200  NEXT I
210  VTAB 22: PRINT "CONTINUI (Y/N)?";
220  GET C$: IF C$ <  > "Y" AND C$ <  > "N" THEN 220
230  IF C$ = "Y" THEN  GOTO 80
240  END
990  REM
991  REM ****    SUBROUTINE 1000    ****
992  REM VISUALIZ. INTESTAZIONI
993  REM
1000  INVERSE
1020  FOR I = 1 TO 3
1030  HTAB 3 + I * 10: PRINT "COLONNA";
1040  NEXT I
1050  PRINT
1060  FOR I = 0 TO 2
1070 S% = 0: IF C% + I < 10 THEN S% = 1
1080  HTAB 18 + I * 10: PRINT  SPC( S%);C% + I;
1090  NEXT I
1100  PRINT
1110  FOR I = R% TO R% + 9
1120 S% = 0: IF I < 10 THEN S% = 1
1130  HTAB 4: PRINT "RIG";: HTAB 8: PRINT  SPC( S%);I
1140  NEXT I
1150  NORMAL : RETURN
```

Caricate questo programma nel vostro calcolatore ed eseguitelo.

All'inizio il programma dedica alcuni secondi al caricamento della matrice X% e in questa fase vi avvisa con un messaggio (istruzioni dalla 30 alla 70). È una buona regola informare l'operatore con un messaggio di avviso quando il calcolatore sembra inattivo perchè esegue delle routine molto lunghe.

Notate che in ingresso viene chiesto il numero di colonna al massimo fino a 12 perchè le altre due saranno senz'altro visualizzate. Analogamente il massimo numero per le righe è 41 perchè in tal caso vengono visualizzate anche le righe 41 e 50.

I valori interi della matrice X% sono convertiti in stringhe alla linea 170 prima di essere stampati. Questa conversione in stringhe ci permette di rendere più semplice la stampa e l'allineamento dei valori. Non è molto facile, infatti, allineare direttamente valori numerici. Provate, per esempio, a modificare così la linea 170:

```
170 PRINT SPC (7); X% (J,I)
```

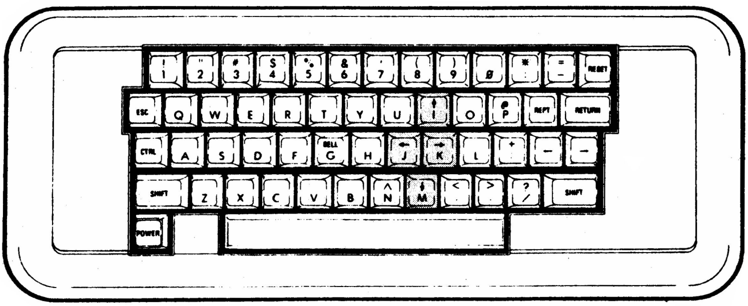
I numeri saranno allineati purchè non visualizzate alcun numero di quattro cifre; in caso contrario le 40 colonne dello schermo non sarebbero sufficienti.

Nel nostro programma abbiamo posta molta attenzione a terminare le visualizzazioni entro la 39esima colonna e non sulla 40esima perchè correremmo il rischio di essere coinvolti nella tecnica di continuazione delle righe detta ad *avvolgimento* (*wrap around*). In tal caso una riga successiva è vista come la continuazione di una riga precedente e vi sarebbe quindi una gran confusione tra i ritorni a capo, da noi voluti, e quelli forzati dalla tecnica di avvolgimento.

Per esercizio provate a modificare il programma in modo da utilizzare anche la 40esima colonna. Per ottenere questo dovete cambiare alla linea 150 la tabulazione orizzontale da 10 a 11; alla linea 1030 da 4+I * 10 in 5+I * 10; alla linea 1080 da 18+I * 10 in 19+I * 10 e infine alla linea 1130 da 4 e 8 in 5 e 9. Se ora eseguite il programma vedrete che tutta la tabella sullo schermo viene alterata da molti ritorni a capo non voluti. Provate ad eliminare questi ritorni a capo indesiderati, ma vi avvisiamo subito che questo risultato non è facile da ottenere.

Notate che, dopo le istruzioni d'ingresso alle linee 80, 100 e 220, sono stati posti dei controlli sulla validità dei dati forniti.

Un miglioramento molto utile, in un programma che visualizza una finestra di dati, consiste nel dare la possibilità all'operatore di potere muovere la finestra stessa a destra e a sinistra, in alto e in basso sullo schermo. Questo è ottenuto mediante i tasti I, J, K e M in modo analogo a quanto visto per gli stessi tasti nel capitolo 3 per lavorare in «edit mode». Il tasto I muove la finestra di una riga in alto, M di una riga in basso, J di una colonna a sinistra e K di una colonna a destra.



Per ottenere questa possibilità, nel nostro programma, sostituiamo le linee dalla 210 alla 240 nel modo seguente:

```
210   VTAB 22: PRINT "CONTINUI?"
215   PRINT "DARE LA DIREZIONE (I,J,K,M),Y/N ";
220   GET C$
225   REM  UNA RIGA IN BASSO?
230   IF R% > 1 THEN  IF C$ = "M" THEN R% = R% - 1: GOTO 120
235   REM  UNA RIGA IN ALTO?
240   IF R% < 41 THEN  IF C$ = "I" THEN R% = R% + 1:GOTO 120
245   REM  UNA COLONNA A SINISTRA?
250   IF C% > 1 THEN  IF C$ = "J" THEN C% = C% - 1: GOTO 120
255   REM  UNA COLONNA A DESTRA?
260   IF C% < 12 THEN  IF C$ = "K" THEN C% = C% + 1: GOTO 120
270   IF C$ = "Y" THEN  GOTO 80: REM INGRESSO
      NUOVA RIGA/COLONNA
280   IF C$ = "N" THEN  END
285   REM  SUONA CAMPANELLO E ATTENDE ALTRO INGRESSO
290   PRINT  CHR$ (7);: GOTO 220
```

Notate che ogni valore in ingresso, diverso da uno dei sei caratteri consentiti, viene rifiutato. Un comando di spostamento della finestra viene anche rifiutato se tenta di spostarla oltre gli estremi della tabella.

PROGRAMMAZIONE DELLE STAMPANTI

Il calcolatore Apple II vede una stampante come un sostituto dello schermo. Per ottenere delle stampe in uscita è quindi necessario porre nel programma delle istruzioni che commutino, l'uscita del calcolatore, dallo schermo alla stampante. Alla fine è necessario poi ripristinare il normale collegamento verso lo schermo mediante l'istruzione PR#.

Le stampanti possono essere collegate al calcolatore tramite interfacce o di tipo parallelo o di tipo seriale a seconda del modello di stampante.

Normalmente le interfacce tipo seriale sono inserite nello slot numero 1 del calcolatore Apple II, mentre quelle tipo parallelo sono inserite nello slot numero 2. Questa scelta è più che altro una convenzione perchè le interfacce potrebbero essere inserite in qualunque slot escluso il numero 0.

Uscita di un testo su una stampante

Se state lavorando con il DOS, il comando PR# viene riconosciuto come una istruzione DOS per cui dovete stamparlo con il carattere CTRL-D come prefisso (il codice ASCII di CTRL-D è 4). Il programma seguente stampa due righe di testo su una stampante, collegata allo slot 1, sotto il controllo del DOS.

```
10  REM USCITA DI 2 LINEE DI TESTO SU UNA STAMPANTE
20  REM CREAZIONE CARATTERE CTRL-D
30 D$ = "": REM  "CTRL-D"
40  REM SELEZIONA PORTA SERIALE I/O
50  PRINT D$;"PR#1"
60  PRINT "QUANDO SCRIVIAMO SU UNA STAMPANTE"
70  PRINT "I DATI ESCONO UN BYTE ALLA VOLTA"
80  REM SCONNETTE LA STAMPANTE
90  PRINT D$;"PR#0"
100  END
```

L'istruzione alla linea 30 crea il carattere di controllo per convertire il comando PR# in una istruzione BASIC. L'istruzione PR# appare alle linee 50 e 90; la prima commuta l'uscita verso la stampante mentre la seconda la riporta sullo schermo. Nelle linee 60 e 70 è contenuto il testo di stampa che apparirà cosi dopo l'esecuzione del programma:

```
QUANDO SCRIVIAMO SU UNA STAMPANTE
I DATI ESCONO UN BYTE ALLA VOLTA
```

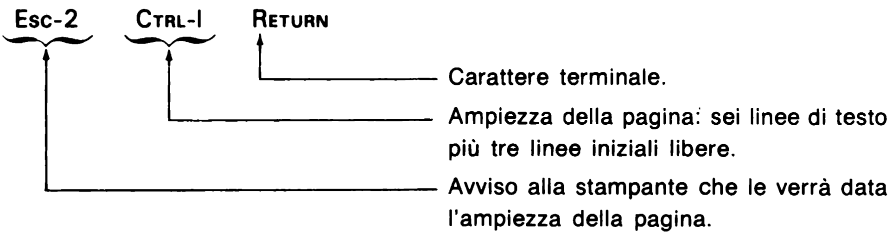
Se il calcolatore non opera con il DOS, il programma è allora il seguente:

```
10  REM USCITA DI 2 LINEE DI TESTO SU UNA STAMPANTE
40  REM SELEZIONA PORTA SERIALE I/O
50  PRINT "PR#1"
60  PRINT "QUANDO SCRIVIAMO SU UNA STAMPANTE"
70  PRINT "I DATI ESCONO UN BYTE ALLA VOLTA"
80  REM SCONNETTE LA STAMPANTE
90  PRINT "PR#0"
100  END
```

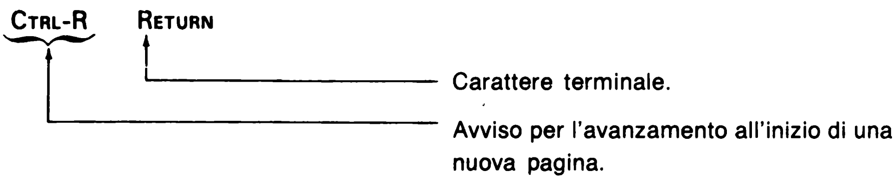
Stampanti programmabili

Molte stampanti offrono la possibilità di eseguire delle stampe, con formati particolari, sotto il controllo dello stesso programma. Questo è possibile inserendo dei caratteri di controllo nello stesso testo che viene inviato alla stampante. In altre parole è possibile variare la lunghezza della linea, cambiare il tipo di caratteri, variare la lunghezza della pagina come anche variare altri parametri di stampa.

La maggior parte delle stampanti, presenti sul mercato dei calcolatori, è collegabile all'Apple II che le gestisce però tutte nella stessa maniera. Supponiamo per esempio di avere una stampante modello 810 della Texas Instruments collegata al nostro calcolatore con una interfaccia seriale allo slot 1. Il costruttore di questa stampante dice che possiamo porre sei righe per pagina mediante la seguente sequenza di caratteri:



Con quest'altra sequenza è invece possibile fare saltare la stampante all'inizio di una pagina nuova:



Per esercizio modifichiamo l'esempio precedente in mdo che le stesse due linee di testo vengano ripetute 15 volte in blocchi di sei linee. Il programma deve girare sotto il controllo del DOS:

```
10  REM  USCITA TESTO SU STAMPANTE
11  REM  CON 6 RIGHE/PAGINA E 5 PAGINE
20  REM  CREAZIONE CARATTERE CTRL-D
30 D$ = "": REM  "CTRL-D"
40  REM  SELEZIONA PORTA SERIALE I/O
50  PRINT D$;"PR#1"
51  REM  SELEZIONA 6 RIGHE/PAGINA
52  PRINT  CHR$ (27);"2";CHR4(9)
53  PRINT  CHR$ (18);
54  FOR I = 1 TO 15
60  PRINT "PROVA DI STAMPA"
70  PRINT "TUTTO BENE"
75  NEXT I
80  REM  SCONNETTE LA STAMPANTE
90  PRINT D$;"PR#0"
100  END
```

Alla linea 52 vengono imposte sei righe per pagina; infatti CHR$(27) rappresenta il carattere ESC, mentre CHR$(9) rappresenta CTRL-I che definisce una pagina di sei linee più tre linee libere. I punti e virgola servono per concatenare i diversi comandi.

L'istruzione alla linea 53 effettua un avanzamento all'inizio pagina. CHR$(18) rappresenta appunto CTRL-R. Il carattere punto e virgola viene posto per evitare di avere subito un ritorno a capo e quindi una linea vuota. Senza di esso la prima pagina avrebbe una linea vuota e cinque linee scritte, mentre le altre pagine avrebbero sei linee scritte.

In questa maniera è possibile programmare tutte le funzioni speciali di una stampante.

Attenzione però che questo programma non può funzionare in Integer BASIC perchè usa la funzione CHR$. È possibile ovviare a ciò inserendo i caratteri speciali di comando in una stringa. Per esempio "ESC" equivale a CHR$(27). Nell'Appendice I potete trovare una lista completa di queste equivalenze.

Stampa di listati di programma

Il comando LIST fa listare sullo schermo qualunque programma contenuto nella memoria del calcolatore. Per ottenere invece la stampa di un programma è necessario anteporre a LIST il comando PR# (al termine bisogna poi ripetere PR#). Supponiamo che la stampante sia collegata tramite una interfaccia nello slot 1, la procedura per stampare un listato è la seguente:

1. Assicurarsi che il programma sia presente nella memoria del calcolatore.
2. Con il cursore su una linea dello schermo, battere il comando PR#1. Il cursore ritorna sulla prima posizione della stessa linea, ma non va a capo.

3. Battere LIST; il comando appare sulla stampante, seguito dal listato del programma che non va più sullo schermo.

4. Al termine della stampa del listato, battere PR#0 (il comando apparirà sulla stampante).

REGISTRAZIONE DI DATI SU CASSETTA

Abbiamo già visto come sia possibile memorizzare un programma su una cassetta per poterlo rileggere in un secondo tempo. In Applesoft è possibile memorizzare su cassetta anche dati numerici.

L'istruzione STORE memorizza variabili con indici su cassetta; l'istruzione RECALL rilegge questi dati e li trasferisce nalla memoria del calcolatore. Ambedue queste istruzioni non comandano il movimento del nastro della cassetta, ne avvisano l'operatore con opportuni messaggi.

Il programma seguente illustra l'impiego di STORE e RECALL. Dapprima vengono assegnati dei valori numerici ad un vettore, poi questo vettore viene memorizzato. Successivamente il vettore viene azzerato ed infine riletto dalla cassetta. Nei punti principali del programma il vettore viene visualizzato per mostrare i suoi cambiamenti.

```
10  REM ******************************
15  REM DIMOSTRAZIONE DELLE ISTRUZIONI
16  REM 'STORE' E 'RECALL'
20  REM ******************************
30  DIM A(10)
40  HOME
50  PRINT  TAB( 4);"MEMOR."; TAB( 13);"CANCEL.";
    TAB( 22);"RICHIAM."
60  REM INIZIALIZ. VALORI ARRAY
70  FOR I = 1 TO 10:A(I) = I: NEXT I
80 T = 8: GOSUB 1000: REM  VISUALIZ. DATI
90  VTAB 20: HTAB 1
100  PRINT "PORRE CASSETTA NEL REGISTRATORE"
101  PRINT "E RIAVVOLGERLA"
110  PRINT "PREMERE TASTI 'RECORD' E 'PLAY'"
120  INPUT "BATTERE 'VAI' ";C$
130  IF C$ <  > "VAI" THEN  GOTO 90
140  STORE A
160  CLEAR : REM AZZERA VALORI DELL'ARRAY
170  VTAB 2:T = 18: GOSUB 1000
180  VTAB 20: HTAB 1
190  GOSUB 1100: REM CANCELLA ULTIMA ISTRUZIONE
200  VTAB 20: HTAB 1
210  PRINT "RIAVVOLGERE NASTRO. PREMERE 'PLAY'"
220  INPUT "BATTERE 'VAI' ";C$
230  IF C$ <  > "VAI" THEN  GOTO 200
240  RECALL A
```

```
260  VTAB 2:T = 28: GOSUB 1000: REM VISUALIZ. DATI LETTI
265  VTAB 20: HTAB 1
270  GOSUB 1100: REM  CANCELLA ULTIMA ISTRUZIONE
280  VTAB 20: HTAB 1
290  PRINT "PREMERE 'STOP'"
300  END
990  REM ****   SUBROUTINE 1000   ****
991  REM VISUALIZ. VALORI DELL'ARRAY
1000  FOR I = 1 TO 10: HTAB T: PRINT A(I): NEXT I: RETURN
1090  REM ****   SUBROUTINE 1100   ****
1091  REM CANCELLA 3 RIGHE DISPLAY
1100  FOR I = 1 TO 119: PRINT " ";: NEXT I: RETURN
```

Il nome della variabile che memorizzate può essere diverso da quello che usate in lettura, ma in genere le due variabili con indici devono avere lo stesso dimensionamento. Nel capitolo 8 vedremo che vi sono alcune eccezioni a questo proposito, ma per il momento vi consigliamo di usare sempre la stessa dimensione, sia in registrazione che in lettura, per le variabili poste in STORE e poi in RECALL.

OTTIMIZZAZIONE DI UN PROGRAMMA

In termini molto generali si può dire che il programma migliore per risolvere un certo problema, sia quello che viene eseguito nel tempo più breve ed occupa meno memoria possibile. Ovviamente non bisogna spingere oltre certi limiti queste caratteristiche perchè un programma deve pur sempre essere affidabile, facilmente leggibile, semplice da usare ed anche modificabile.

Sebbene sia preferibile però che un programma soddisfi in primo luogo queste ultime caratteristiche, è senz'altro raccomandabile che sia anche veloce nella sua esecuzione ed occupi poca memoria. Un programma di questo tipo ha sicuramente una struttura molto semplice a tutto vantaggio della sua efficienza.

Se ora cerchiamo di comprendere che cosa rende un programma rapido e che cosa gli fa risparmiare memoria, ci accorgiamo che questi due requisiti sono spesso in antagonismo per cui sarà lasciato al programmatore di stabilire, di volta in volta, il giusto compromesso tra velocità di esecuzione ed occupazione di memoria.

PROGRAMMI RAPIDI

Vediamo alcune tecniche per rendere veloce un programma.

Evitare di usare le costanti (per esempio 3.14, "INGRESSO"), ma assegnate invece il loro valore a delle variabili. In altre parole ponete delle variabili eguali alle costanti subito all'inizio del programma e poi usate le variabili al posto delle costanti. Questo è molto importante quando dovete usare costanti intere in espressioni reali. Il calcolatore richiede molto più tempo, per convertire un intero in reale, che a leggere il valore di una variabile. Noterete il vantaggio quando queste costanti intere siano

presenti in un ciclo FOR-NEXT o in una subroutine molto usata o in una funzione definita dall'utente. Queste assegnazioni iniziali presentano anche il vantaggio di potere cambiare le costanti in una istruzione sola, e non in molti punti del programma, nel caso dobbiate appunto modificarle.

Richiamate il prima possibile quelle variabili che userete più spesso nel corso del programma. Il BASIC pone le variabili, nell'area di memoria loro riservata, mano mano che le incontra durante l'esecuzione del programma. Per questo motivo è preferibile che all'inizio di questa area siano presenti quelle variabili che vengono usate più spesso.

Quando il BASIC incontra una istruzione di salto, esso cerca sequenzialmente, a partire dall'inizio del programma, la linea a cui saltare. Di conseguenza i salti a linee con numerazione bassa vengono eseguiti in minor tempo. Se dovete quindi fare spesso dei salti, come nel caso di subroutine molto usate, ponete tali subroutine all'inizio del programma.

In Applesoft non indicate la variabile di ciclo nell'istruzione NEXT. La chiusura di un ciclo è sempre univoca per cui potete risparmiare il tempo di verifica se la variabile posta è quella giusta.

PROGRAMMI COMPATTI

Usate le subroutine tutte le volte che vi è possibile. Questo rende il programma anche più facile da leggere e da modificare.

Usate gli indici zero delle variabili con indici (cioè i valori X(0), B(0)).

Se una costante ha molte cifre allora è preferibile assegnarla ad una variabile con un nome corto, cioè con pochi caratteri, e usare poi la variabile dove serve.

Ponete più istruzioni possibile su una stessa linea. Ogni linea di programma, con sua numerazione, occupa infatti cinque byte di memoria in più. Attenzione però che linee molto lunghe sono difficili da gestire, cioè da scrivere e da leggere.

Ponete un limitato numero di commenti REM possibilmente abbreviati. Non riducete troppo però i commenti perchè un programma deve sempre essere intelleggibile con facilità.

Ponete molta attenzione nell'uso di nuove variabili. Qualunque variabile, anche se richiamata una sola volta, occupa un certo spazio di memoria. Per minimizzare l'uso delle variabili potete riservare alcuni nomi per variabili di servizio che ogni volta dovete però inizializzare. Queste variabili sono per esempio quelle contenute solamente nei cicli FOR-NEXT oppure quelle che fanno parte di calcoli intermedi. Per evitare confusioni, stabilite voi stessi una regola per denominare le variabili, come per esempio NC$ per il nome del cliente e V$ per il venditore, e usate invece la X come primo carattere di tutte le variabili di servizio.

Quando vi è possibile non usate l'istruzione DATA, o le normali assegnazioni, per fornire i dati al vostro programma, ma usate invece l'istruzione INPUT o accedete direttamente ai file di dati (se avete i dischi).

In Applesoft usate vettori e matrici interi e non reali quando vi sia possibile. Infatti

gli elementi interi occupano solo due byte, mentre quelle reali ne occupano cinque.

Usate frequentemente la funzione FRE, nel vostro programma, per annullare l'area di memoria delle stringhe.

CORREZIONE DEGLI ERRORI – DEBUGGING

Ben difficilmente un nuovo programma funziona subito correttamente. Anche se non vengono segnalati errori di sintassi, è molto facile che vi siano errori di logica nella sua struttura. Nella terminologia dei calcolatori si è sempre usata una parola che rende molto bene il concetto di correggere un programma: si dice infatti *spulciare* un programma (in inglese *debugging).* Diciamo subito che questo è il momento in cui si distingue un programmatore prefessionista da un dilettante.

In ogni caso esistono alcune istruzioni che permettono di facilitare il compito di correzione.

L'istruzione PRINT

Inserite, momentaneamente, nel programma delle istruzioni PRINT in tutti quei punti che ritenete più opportuni per farvi stampare dei valori "strategici". Potete così controllare se il vostro programma si comporta come avevate previsto. Fatevi visualizzare sia valori di variabili come anche messaggi di avviso (per sapere per esempio se un certo salto è stato fatto oppure no).

L'istruzione TRACE

Questa istruzione permette di seguire passo passo l'esecuzione di ogni singola istruzione in quanto visualizza il numero di ogni linea eseguita. Per vedere come funziona caricate questo programma poi battete TRACE e RUN:

```
100  PRINT "ENTER A NUMBER FROM 1 TO 5 (6 TO END)";
110  INPUT N
120  IF N = 1 THEN  PRINT "UNO";
130  IF N = 2 THEN  PRINT "DOS";
140  IF N = 3 THEN  PRINT "TRES";
150  IF N = 4 THEN  PRINT "CUATRO";
160  IF N = 5 THEN  PRINT "CINCO";
170  IF N > 5 THEN  GOTO 100
180  FOR I = 1 TO N
190  PRINT " *";: REM  PRINT N ASTERISKS
200  NEXT I
210  CALL  - 936: REM  CLEAR SCREEN
220  GOTO 100
```

Per annullare il comando TRACE dovete battere NO TRACE.

L'istruzione DSP

L'istruzione DSP è disponibile solo in Integer BASIC. Ecco un esempio:

```
>10 DSP COUNT
```

L'istruzione DSP fa si che l'operatore sia avvisato di ogni cambiamento di valore della variabile COUNT e a quale linea esso è avvenuto. Dal momento che RUN disabilita il comando DSP, per eseguire il programma dovete usare GOTO oppure porre DSP nel programma stesso.

Per annullare DSP potete dare il comando contrario NO DSP.

```
>300 NODSP NAME$
```

In questo caso non vengono più segnalati i cambiamenti della variabile NAME$.

Per vedere gli effetti di DSP aggiungete queste linee all'esempio che abbiamo usato per l'istruzione TRACE; successivamente provate a togliere il comando TRACE per vedere i soli effetti di DSP:

```
10 DSPN
20 DSPI
100  PRINT "ENTER A NUMBER FROM 1 TO 5 (6 TO END)";
110  INPUT N
120  IF N = 1 THEN  PRINT "UNO";
130  IF N = 2 THEN  PRINT "DOS";
140  IF N = 3 THEN  PRINT "TRES";
150  IF N = 4 THEN  PRINT "CUATRO";
160  IF N = 5 THEN  PRINT "CINCO";
170  IF N > 5 THEN  GOTO 100
180  FOR I = 1 TO N
190  PRINT " *";: REM  PRINT N ASTERISKS
200  NEXT I
210  CALL  - 936: REM  CLEAR SCREEN
215 NODSPN
220  GOTO 100
```

RESTRIZIONI TRA MODO DIFFERITO E MODO IMMEDIATO

La maggior parte delle istruzioni BASIC possono essere usate sia in modo differito che in modo immediato. In alcuni casi però è possibile usarle solo in uno dei due modi per cui nelle tabelle seguenti riportiamo queste limitazioni. La Tabella 4.2 si riferisce all'Integer BASIC, mentre la Tabella 4.3 si riferisce all'Applesoft.

Tabella 4.2 — Istruzioni dell'Integer BASIC che si possono usare solo in modo differito o in modo immediato.

Solo in modo differito	Solo in modo immediato
END	AUTO
FOR	CLR
GOSUB	CON
INPUT	DEL
NEXT	HIMEM:
RETURN	LOAD
	LOMEM:
	MAN
	NEW
	RUN
	SAVE

Tabella 4.3 — Istruzioni dell'Applesoft valide solo in modo differito.

Solo in modo differito
DATA
DEF FN
GET
INPUT
ON ERR GOTO
RESUME

CAPITOLO 5

IL DISCO II

L'unità a disco è una delle componenti più importanti di un sistema di elaborazione. Essa permette di accedere, in tempi molto brevi, a grandi archivi di dati. L'unità Disco II del sistema Apple può memorizzare, per esempio, 100.000 caratteri su un solo dischetto. E ciò rappresenta più del doppio di una memoria RAM di 48K. Inoltre i dischetti mantengono le informazioni registrate anche quando il calcolatore viene spento a differenza delle memorie RAM che invece le perdono.

CARATTERISTICHE DEI DISCHI

La registrazione dei dati sui dischi avviene in forma magnetica come per i nastri e le cassette magnetiche. La maggiore differenza rispetto questi ultimi, riguarda la forma stessa del disco che è rotonda ed il fatto che il disco è sempre tenuto in rotazione. La lettura e la registrazione dei dati avviene tramite una testina magnetica contenuta nel drive. Il calcolatore comanda lo spostamento di questa testina su qualunque punto della superficie del disco. Questa caratteristica viene definita *accesso random* (o *accesso diretto*). Per questo motivo si dice che il disco è un dispositivo di memoria ad accesso random. Un programma speciale, chiamato *Sistema Operativo a Disco* (*DOS*), controlla tutte le operazioni sul disco. Dal punto di vista fisico esistono vari tipi di disco.

Dischi rigidi

Questi dischi sono appunto rigidi come dice il loro nome (in inglese sono chiamati hard disk); normalmente possono contenere dai 5 ai 10 megabyte (1 megabyte = 1 milione di byte). La maggior parte dei dischi rigidi sono removibili, cioè il disco e il drive sono separabili. Questo permette di usare più dischi, intesi come supporto, con uno stesso drive. Un disco rigido può costare dalle 150.000 lire, mentre un drive costa dai 3 ai 10 milioni di lire. In Figura 5.1 è riportato un classico esempio di disco rigido.

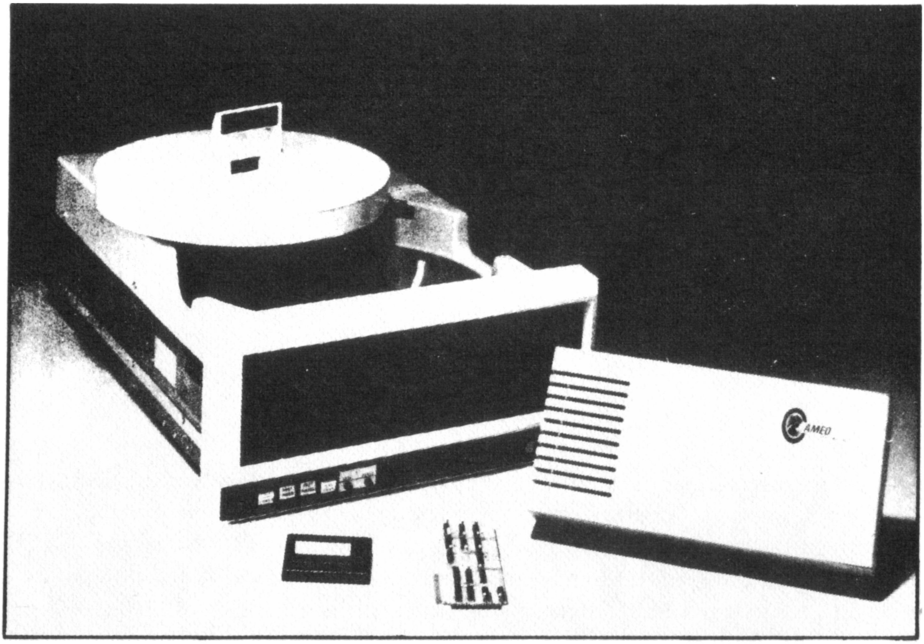


Figura 5.1 — Esempio di disco rigido.

Dischi Winchester

Un esempio di disco Winchester appare in Figura 5.2. Questo disco usa una speciale tecnologia per cui può contenere da sei a dieci volte più dati rispetto un altro tipo di disco. I dischi Winchester devono essere assolutamente protetti dalla polvere (perfino il fumo delle sigarette li può danneggiare!) per cui sono contenuti in cartucce speciali ermetiche. Essi non sono removibili dal drive. Possono costare dai 3 agli 8 milioni di lire.

Dischetti

I dischetti sono ormai il tipo più comune di disco. Essi sono costituiti da un supporto flessibile contenuto in una busta di plastica quasi rigida. Questa busta protegge sempre il dischetto anche quando è caricato nel drive. Il dischetto può infatti ruotare liberamente nella busta. Non estraete mai un dischetto dalla sua busta! In Figura 5.3 appare un dischetto tolto dalla sua busta.

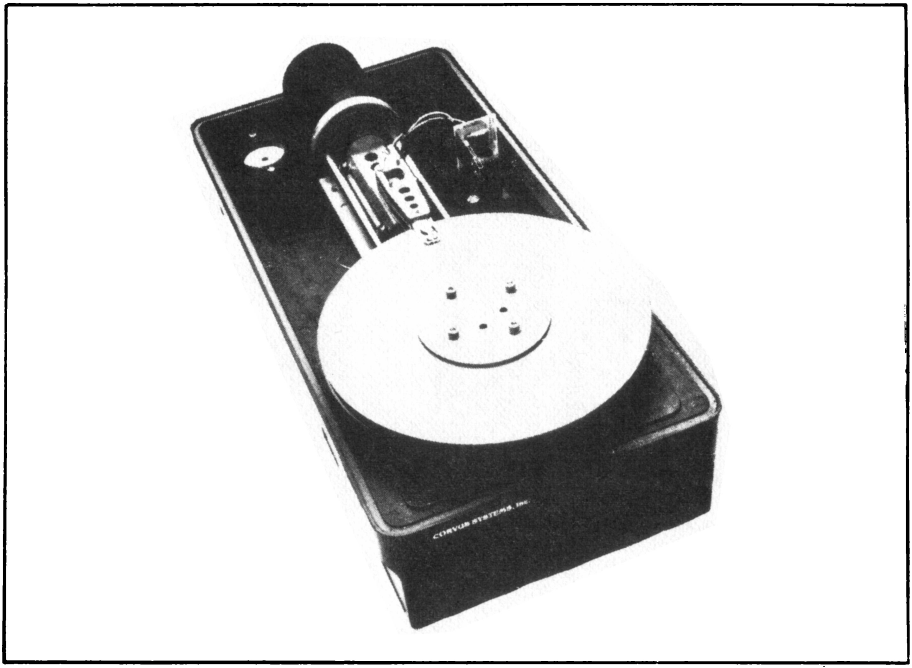


Figura 5.2 — Disco tipo Winchester.

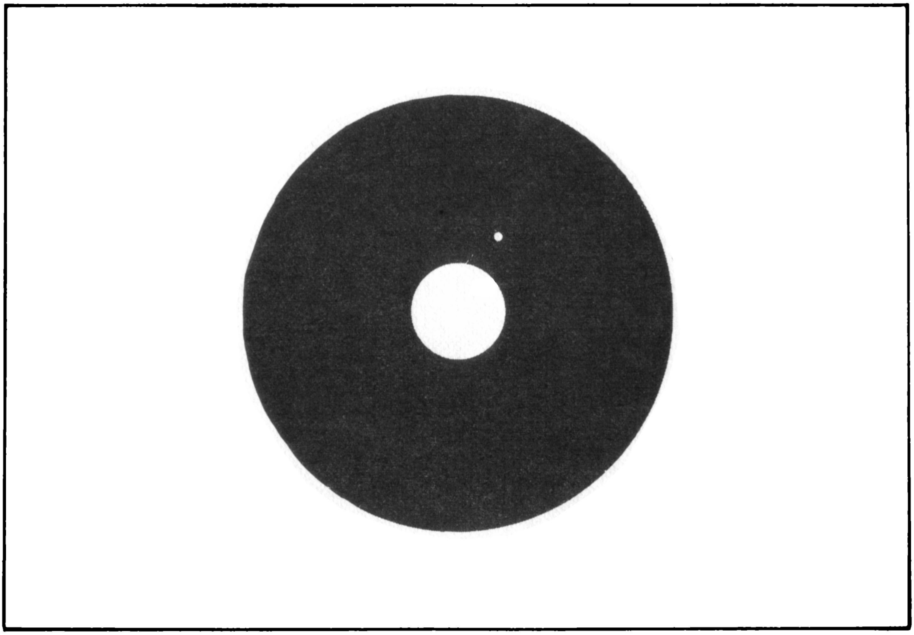


Figura 5.3 — Dischetto flessibile senza busta protettiva.

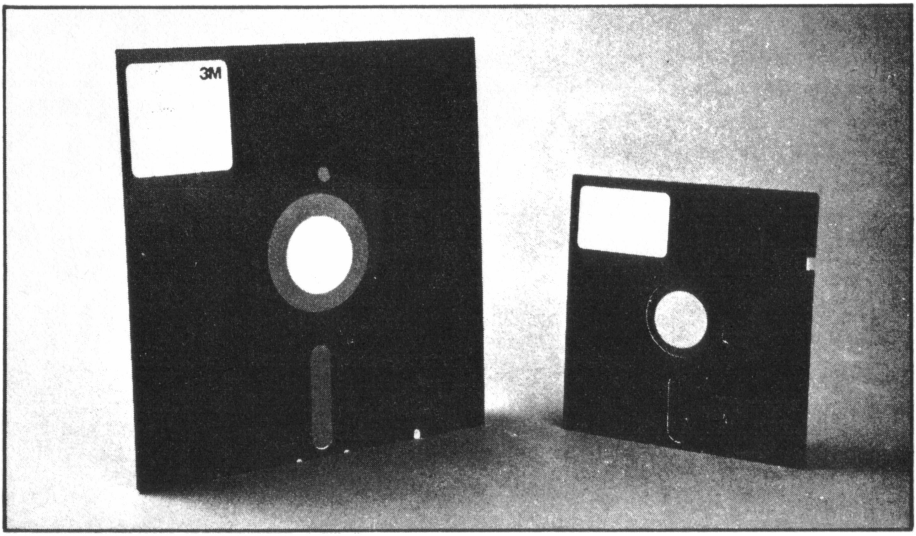


Figura 5.4 — Dischetti a 8 pollici e 5-1/4 pollici.

I dischetti sono anche chiamati *floppy disk* e possono avere due dimensioni: diametro di 8 pollici oppure di 5-1/4 pollici. In Figura 5.4 sono riportati ambedue. I dischi nella versione più piccola sono anche chiamati *mini-dischi* o *mini-dischetti* o *mini-floppy.* Il Disco II dell'Apple usa appunto questi dischetti e può memorizzare su ognuno di essi fino a 100.000 byte.

COME I DATI SONO REGISTRATI SUL DISCO

Cerchiamo di illustrare i concetti fondamentali della registrazione dei dati su un disco.

Tracce e Settori

I 100.000 byte memorizzati su un dischetto sono ripartiti su 35 *tracce* concentriche. In un certo senso le tracce assomigliano ai solchi di un disco musicale, ma con la differenza che esse sono su circonferenze concentriche e non su una spirale. Le tracce sono numerate da 0 a 34 (Figura 5.5).

Per individuare quindi uno dei 100.000 byte registrati su un dischetto il DOS deve accedere prima alla traccia, poi al settore e infine riconoscere quale dei 256 byte di questo settore è quello voluto. Ricordiamo infatti che il disco è una unità ad accesso diretto per cui si può leggere direttamente il settore voluto (Figura 5.6).

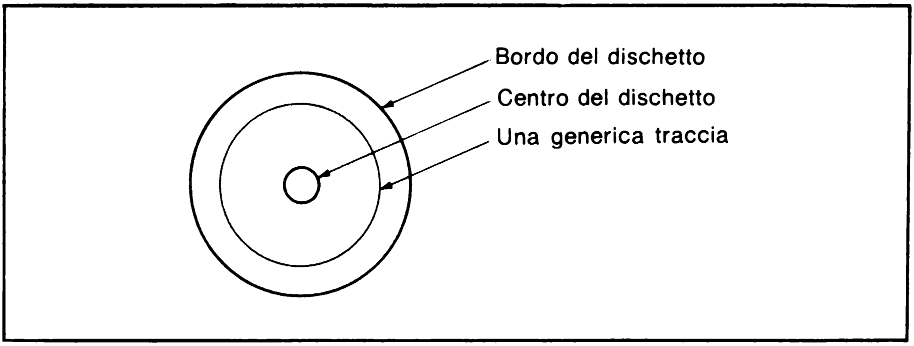


Figura 5.5 — Tracce su un dischetto.

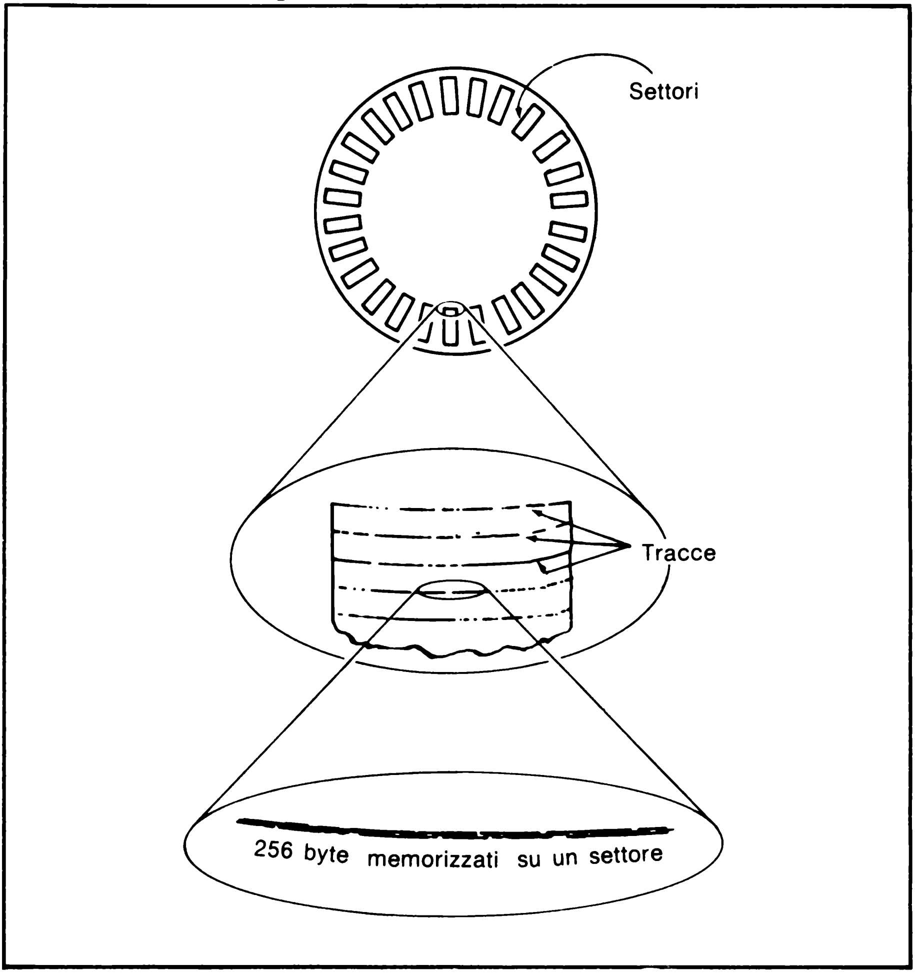


Figura 5.6 — Superficie di un dischetto registrata.

ACCESSO ALLE TRACCE E AI SETTORI

Per accedere ad una determinata traccia il calcolatore fa spostare la testina magnetica proprio come voi sposterete il pick-up di un giradischi per sentire una certa canzone su un disco musicale.

Nel caso invece dei settori, i metodi di accesso sono più complicati. In genere esistono due tecniche per individuare un settore ed ambedue fanno uso di almeno un *foro indice*. Tutti i dischetti hanno sempre un foro vicino al centro sulla busta a cui corrisponde almeno un foro sul dischetto interno. Se quindi esiste un solo foro sul dischetto interno allora si dice che il dischetto è *settorializzato a software;* se invece esistono più fori, uno per ogni settore, allora il dischetto è *settorializzato ad hardware.* Per capire il funzionamento di questi fori basta pensare che nel drive vi è una lampadina ed una cellula fotoelettrica mediante le quali vengono rilevati o tutti i settori o almeno il loro inizio.

Settori hardware

Come abbiamo già detto esiste un foro in corrispondenza di ogni settore (Figura 5.7) più un ulteriore foro per individuare l'inizio dei settori. Il calcolatore individua i settori contando i fori dopo il primo settore.

Settori software

In questo caso esiste solo un foro che determina l'inizio dei settori (Figura 5.8). La determinazione degli altri settori è fatta a tempo a partire dal momento che viene rivelato il primo settore. Il Disco II dell'Apple usa la settorializzazione a software.

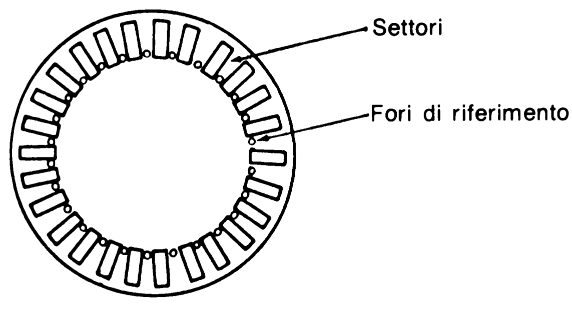


Figura 5.7 — Settorializzazione ad hardware.

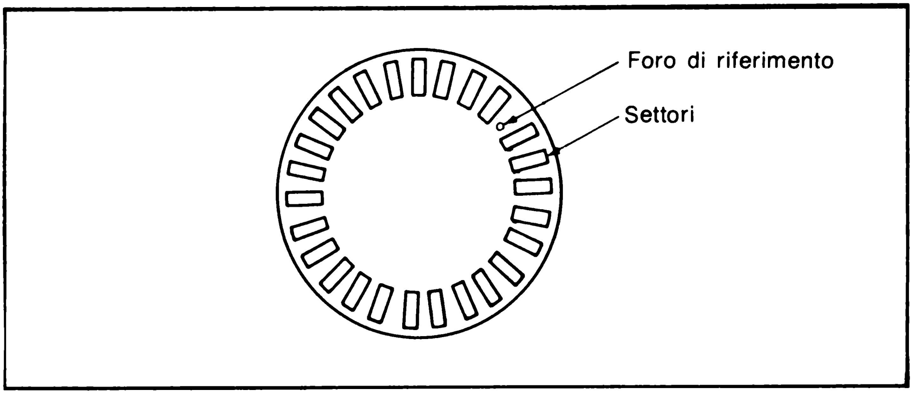


Figura 5.8 – Settorializzazione a software

PROTEZIONE DA SCRITTURA

È possibile interdire la scrittura su un dischetto quando si vuole evitare che erroneamente si scriva su un disco importante. Per ottenere questo è sufficiente coprire, con del nastro adesivo, la tacca apposita che si trova su un lato della busta (Figura 5.9). Attenzione però che questo è valido solo per i dischetti con diametro di 5-1/4 pollici: cioè per impedire la scrittura bisogna coprire la loro tacca. Nel caso invece dei dischetti con diametro di 8 pollici, per impedire la scrittura bisogna lasciare aperta la tacca.

Alcuni dischetti sono sempre protetti contro la scrittura (nel nostro caso sono cioè completamente privi di tacca) perchè contengono dei programmi molto importanti. Il "System Master Diskette", fornito con il Disco II, è appunto uno di questi dischi permanentemente protetti.

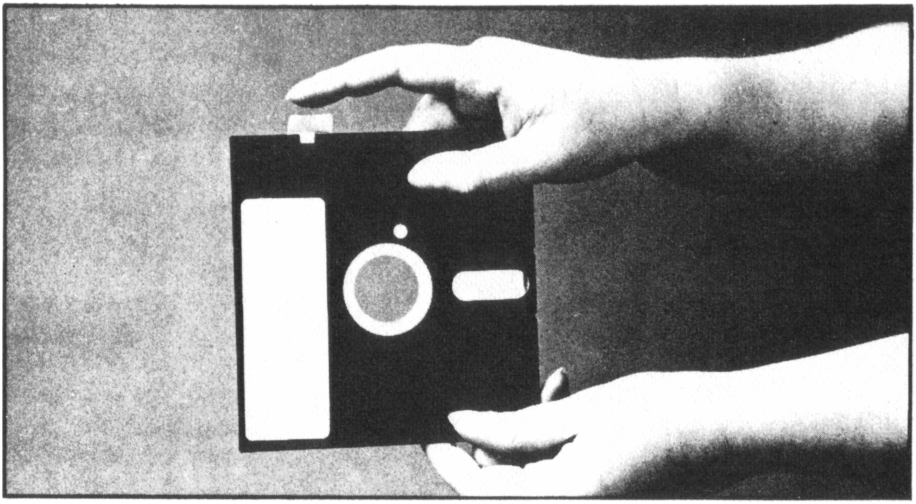


Figura 5.9 – Protezione da scrittura di un dischetto di 5-1/4 pollici.

IL SISTEMA OPERATIVO A DISCO

Tutte le operazioni che impegnano il disco sono controllate da uno speciale programma chiamato *Sistema Operativo a Disco* (DOS). Anche il BASIC quando deve dialogare con il disco, usa il DOS.

VERSIONI DI DOS

Come per tutti i programmi, anche del DOS esistono varie versioni sempre più recenti. Nel nostro caso la versione più recente è il DOS 3.3 che descriviamo in questo capitolo. La versione precedente a questa è il DOS 3.2.1. La differenza tra le due consiste nel fatto che la più recente prevede 16 settori per traccia sul dischetto, mentre la precedente ne prevedeva solo 13. Se necessario è possibile, mediante uno speciale programma, convertire i dischetti dal DOS versione 3.2.1 in quello versione 3.3.

INIZIALIZZAZIONE DEI DISCHI

Prima di poter usare un dischetto con l'Apple II è necessario *inizializzarlo*. Questa procedura fa cancellare tutti i vecchi file sul dischetto e pone una copia del DOS sulle tracce 0, 1 e 2. Le istruzioni per l'inizializzazione saranno date in seguito.

FILE SU DISCO

Le informazioni sono in genere registrate sui dischi nella forma di *file*. Un file può avere qualunque lunghezza purchè possa fisicamente risiedere tutto su un disco. Un file ha sempre un nome e può contenere informazioni che riguardano un testo, un programma, un grafico o un archivio di dati. I vari tipi di file saranno discussi dettagliatamente in seguito.

DIRECTORY DI UN DISCHETTO

Tutti i nomi dei file che risisedono su un dischetto sono sempre elencati nella *directory* del dischetto. La directory è come un indice che permette di risalire ai singoli file. Essa è posta sulla traccia 17; inizia dal settore 15 e termina al settore 1. Nelle nostre directory possiamo elencare sino a 84 file.

Assieme al nome del file viene registrato un codice che indica il tipo di dati del file, il numero di settori occupati e l'indirizzo del primo settore che contiene la *lista delle tracce/settori* per ogni file.

LISTA DELLE TRACCE/SETTORI

La lista delle tracce/settori contiene delle coppie di byte che individuano l'indirizzo – numero di traccia e numero di settore – di ogni settore occupato dal file. Ogni cop-

pia di byte viene chiamata *link*. Il primo link della lista fornisce l'indirizzo di un eventuale secondo settore usato dalla lista stessa. La lista può occupare quanti settori necessita. Il secondo link è invece l'indirizzo del primo settore occupato dal file vero e proprio; il terzo link individua il secondo settore del file e così via. La fine della lista è indicata da un link contenente solo zeri.

Processo di registrazione su disco

Il DOS controlla il trasferimento di tutti i dati verso il disco o dal disco. Ecco quello che succede quando scrivete in un file:

1. Il DOS ricerca nella directory il nome del file.
2. Se il nome viene trovato il DOS legge 256 byte da un certo settore e li memorizza in una zona della memoria detta *buffer*. Se il nome non è trovato, oppure in questo settore non è stato scritto alcun dato in precedenza, il DOS riempie il buffer con degli zeri.
3. I primi 256 byte, che devono essere scritti nel file, sono copiati nel buffer. Se i dati sono meno di 256, allora nel buffer rimangono i vecchi dati.
4. Se esattamente 256 byte sono scritti nel buffer, allora il contenuto del buffer viene trasferito nel settore del disco indicato al punto 2.
5. I punti 3 e 4 vengono ripetuti sino quando tutti i dati da portare su disco sono stati dapprima copiati nel buffer e poi trasferiti sul disco.
6. Dopo che tutti i dati sono stati trasferiti sul disco, il DOS aggiorna la lista delle tracce/settori e la directory.

Notate che se il numero totale di byte da trasferire su disco non è divisibile per 256, allora l'ultimo blocco di dati non riempirà completamente il buffer e non verrà quindi trasferito sul disco. Per questo motivo esiste il comando di *chiusura del file* CLOSE per forzare il trasferimento dell'ultimo blocco e per aggiornare la lista delle tracce/settori e della directory. È molto importante che eseguiate sempre correttamente la chiusura di un file perchè diversamente potreste perdere dei dati.

DISCHI ROVINATI

Parlare di dischi rovinati o deteriorati può significare due diverse cose: i dischi sono *rovinati come software* oppure sono *rovinati come hardware.*

Se un disco è rovinato come hardware allora significa che è danneggiato fisicamente. Cioè può essere graffiato oppure rotto. In questo secondo caso è molto probabile che rovini la testina di lettura/scrittura del drive che a sua volta potrà rovinare altri dischi. Per questo motivo dovete trattare sempre con molta delicatezza i vostri dischetti.

Nell'altro caso diciamo che un disco è rovinato come software se, pur essendo fisi-

camente integro, ha perso delle informazioni privilegiate come quelle riportate nella directory. Questo può succedere se per errore viene scritto qualcosa di spurio sulla directory o sulla lista delle tracce/settori. Spesso inconvenienti di questo genere succedono quando non si chiude correttamente un file in scrittura, poi si toglie il dischetto dal drive e se ne inserisce un altro. Se a questo punto vi ricordate di chiudere il file originale, allora il secondo dischetto viene rovinato come software.

CARICAMENTO DEL DISCO II

Per poter lavorare con i comandi per il disco dovete aver prima caricato il Sistema Operativo a Disco (DOS). Teoricamente, se aveste molto tempo, potreste pensare di caricare il DOS dalla tastiera, ma è ovvio che esistano dei metodi più pratici. Nella terminologia inglese si dice fare il *booting* del disco o del DOS.

COME FARE IL BOOTING DEL DOS

Vi sono vari modi per fare il booting che dipendono dalla configurazione del vostro sistema e dal linguaggio che usate per fare tale caricamento. Tutti i metodi prevedono però che il drive sia connesso al connettore 1 della cartolina controllore inserita nello slot numero 6.

Inserite il dischetto "System Master Diskette" nel drive e chiudete il portellino. I passi successivi dipendono dalla vostra configurazione, ma alla fine dovrete sempre ottenere sullo schermo una delle immagini riportate nella Figura 2.5.

Autostart

Come dice il nome stesso il caricamento del DOS avviene automaticamente. Per potere eseguire questa partenza automatica dovete però avere l'Autostart Monitor. Per controllare se avete questa opzione, installata nel vostro calcolatore, è sufficiente che accendiate il calcolatore con il Disco II collegato. Se la lampadina rossa IN USE del disco si accende, e sentite un ronzio, allora avete l'Autostart Monitor.

Per fare quindi il booting con l'Autostart Monitor è sufficiente che accendiate il calcolatore stesso.

Booting dal Monitor

Dopo che il carattere di pronto del Monitor (*) è apparso sullo schermo, potete dare i comandi dell'Assembly Language Monitor. Per caricare il DOS è possibile usare diverse procedure.

Booting con una istruzione di salto

Battete una lettera C seguita dal numero dello slot a cui è collegato il drive (normalmente il sesto) e poi due zeri e una lettera G. Ecco un esempio:

```
*C600G
```

C600 è l'indirizzo di memoria del programma che esegue il caricamento dallo slot 6. G è il comando per trasferire il controllo a questo programma. Alla fine premete RETURN. La lampadina del drive si deve accendere e udirete un certo ronzio.

Booting con i comandi CTRL-K e CTRL-P del Monitor

Il booting può essere eseguito anche battendo il numero di slot (normalmente 6) e poi uno dei due comandi CTRL-K o CTRL-P seguito da RETURN. Attenzione però che i due comandi CTRL-K e CTRL-P non sono visualizzati sullo schermo.

Booting dall'Integer BASIC o dall'Applesoft

I comandi sono gli stessi sia per l'Integer BASIC che per l'Applesoft.

Booting dal BASIC con i comandi PR# e IN#

Dopo il carattere di pronto (> per l'Integer BASIC e] per l'Applesoft) battete i comandi PR# o IN# seguiti dal numero di slot a cui è collegato il drive. Ecco due esempi:

```
PR#6
```

oppure:

```
IN#6
```

Premete quindi RETURN.

Booting con il Language System

Se avete la cartolina Language System, le procedure precedenti permettono di caricare solamente il DOS versione 3.3. Se lavorate invece con le versioni di DOS 3.2.1, 3.2 o precedenti, dovete usare due dischi.

In questo secondo caso inserite dapprima nel drive il dischetto "BASICS: Integer and Applesoft II" (fornito con il Language System), al posto del dischetto System Master Diskette, e procedete al suo caricamento con una qualunque delle procedure precedenti (autostart, PR#6, ecc.). Alla fine sullo schermo vi appare:

```
INSERT BASIC DISK AND PRESS RETURN
```

Adesso dovete inserire il dischetto System Master Diskette oppure un altro dischetto che sia già stato inizializzato. Dopo aver premuto RETURN il booting procede normalmente e alla fine vi appare sullo schermo una delle immagini della Figura 2.5.

I PRIMI COMANDI PER IL DISCO

Il Sistema Operativo a Disco interpreta ed esegue comandi rivolti al disco. Alcuni di questi comandi richiedono la definizione di opportuni parametri, mentre altri sono molto più semplici.

Iniziamo ad esaminare questi secondi.

CATALOG

Il comando CATALOG fa visualizzare, sullo schermo o sulla stampante, tutti i nomi dei file di un dischetto.

La prima cosa che viene indicata è il *numero di volume* del dischetto di cui parleremo in seguito.

Per ogni file viene indicato il *tipo* di dati, se il file è *locked,* il *numero di settori* occupati dal file e il *nome del file.* Un esempio di catalogo è riportato in Figura 5.10.

Tipo di file

Il codice per i tipi di file è riportato nella Tabella 5.1. La lettera di questo codice appare nel catalogo come prima lettera a sinistra.

File protetti (Locked)

I file protetti sono quelli in cui non è possibile scriverci e non possono essere cancellati. Nel catalogo essi vengono indicati con un asterisco (*) posto a sinistra del codice di tipo. Questa protezione verrà descritta più avanti.

```
*I 002 HELLO
*I 053 APPLE-TREK
*I 018 ANIMALS
*B 009 UPDATE 3.2.1
*I 014 COPY
*I 009 COLOR DEMO
*I 053 BRICK OUT
*I 026 SPACE WAR
*I 050 THE INFINITE NO. OF MONKEYS
*I 051 COLOR SKETCH
*I 053 SUPERMATH
*I 026 APPLEVISION
*I 017 BIORHYTHM
*I 027 PINBALL
```

Figura 5.10 — Un esempio di Catalogo di disco.

Tabella 5.1 — Codice dei tipi di file.

Codice	Significato
A	Programmi Applesoft
B	File in binario
I	Programmi in Integer BASIC
T	File di testo

Numero di settori

Il numero di settori occupati dal file è indicato con tre cifre. Il file più corto (vuoto) occupa un solo settore. Se un file è molto lungo e ha più di 255 settori, allora questo numero riparte da 0. Questo fatto non influenza la vera lunghezza del file.

Nomi dei file

Il DOS dell'Apple II vuole che i singoli file abbiano un nome. Le regole per dare questi nomi sono le seguenti:

1. I nomi di file possono avere da 1 a 30 caratteri. Ulteriori caratteri sono trascurati.
2. I nomi devono iniziare con una lettera.
3. Qualunque carattere, che potete battere alla tastiera, può far parte di un nome di file eccetto le virgole.

Potete usare anche i caratteri non stampabili nel nome di un file (come le sequenze CTRL-), ma essi non appariranno nel catalogo. Questo fatto è molto utile perchè vi permette di dare ai vostri file dei nomi riservati che non saranno mai leggibili sul catalogo; attenzione però a ricordare voi questi nomi!

Uso del comando CATALOG

Per dare questo comando basta introdurlo sotto il controllo del DOS. Battete infatti:

```
CATALOG
```

E otterrete qualcosa di simile alla Figura 5.10.
Se il numero di linee del catalogo è superiore a 18, il calcolatore le visualizzerà e poi attenderà che voi battiate un qualunque tasto (eccetto RESET, CTRL e SHIFT) per visualizzare le altre linee. Questa pausa vi permette di leggere tutti i nomi dei file prima di farli scorrere fuori dallo schermo.

LOAD

Il comando LOAD trasferisce un file di programma dal dischetto nella memoria del calcolatore. Ovviamente dovete specificare quale programma volete mediante il suo nome. Per esempio questo comando carica il programma COLOR DEMOSOFT:

```
LOAD COLOR DEMOSOFT
```

Se il nome non è contenuto nella directory del dischetto, vi viene segnalato il messaggio di errore FILE NOT FOUND.

Se il nome del file è presente sul dischetto, il DOS controlla il tipo di file e se esso non è un programma vi viene segnalato l'errore FILE TYPE MISMATCH.

Se tutto è corretto, il comando LOAD dapprima cancella tutta la memoria del calcolatore poi trasferisce, cioè carica, il nuovo programma in memoria. Alla fine vi appare sullo schermo il carattere di pronto e voi potete quindi listare, eseguire o modificare il vostro programma.

VERSIONE A DISCO DEL COMANDO RUN

Quasi sempre, dopo che avete dato il comando LOAD, darete anche il comando RUN. Per praticità è possibile dare solo questo secondo comando in modo che il primo sia implicito. Cioè potete battere subito RUN seguito però dal nome del programma senza battere LOAD. Ecco alcuni esempi:

```
RUN PROGRAM 2
RUN SPOT RUN
RUN COLOR DEMOSOFT
```

INDICAZIONE DEL NUMERO DI DRIVE

Molti comandi DOS prevedono che possiate specificare il numero di drive. I parametri necessari sono due: quello relativo al *drive* e quello relativo allo *slot.*

Ad un controllore per disco possono sempre essere collegati due drive. Questi due drive sono indicati dai parametri D1 e D2. D1 o D2 possono quindi essere aggiunti ai comandi LOAD e RUN:

```
LOAD UP,D2
RUN AROUND,D1
CATALOG,D2
```

Dopo che avete specificato un drive, in una istruzione per il disco, il calcolatore adotterà sempre lo stesso drive implicitamente (default). Solo con un'altra indicazione potete cambiare drive. Se il parametro di drive non viene specificato il calcolatore usa sempre il valore D1.

INDICAZIONE DELLO SLOT

La cartolina di controllo del Disco II, come abbiamo già spiegato, viene inserita in uno slot del calcolatore. Nel calcolatore Apple II esistono otto slot, ma lo slot numero 0 non può essere usato per i dischi. Rimangono quindi disponibili sette slot per i dischi. Ogni cartolina di controllo può servire due drive per cui, in definitiva, si possono collegare 14 drive ad un calcolatore Apple II.

Se possedete più di due drive non pensiate di poterli denominare D3, D4, ecc., ma dovete usare sempre D1 e D2 e aggiungere il numero di slot S1, S2,... S7. Ecco un esempio:

```
CATALOG,S5
LOAD TRUCKS,S6
RUN OVER,S3
```

Il numero di slot, usato facendo il booting del DOS, diviene il valore implicito (default) per lo slot. Solo con una indicazione esplicita potete cambiare tale numero di slot.

In uno stesso comando per disco potete indicare ambedue i parametri di drive e di slot: il primo da 1 a 2, il secondo da 1 a 7. I due parametri possono essere dati con qualunque ordine. Ecco alcuni esempi:

```
CATALOG,D2,S5
CATALOG,S5,D2
```

Questi due esempi sono equivalenti e fanno apparire sullo schermo il catalogo del dischetto posto nel drive 2 del controllore posto nello slot 5.

Inconvenienti con il numero di slot

Se date un numero di slot, che non contiene alcun controllore, il calcolatore entra in attesa del controllore che non esiste e rimane indefinitivamente in questo stato. Per uscire da questa situazione premete RESET. Se ottenete in risposta il carattere di pronto del vostro BASIC, allora tutto è corretto. Se invece ottenete il carattere di pronto del Monitor (*), allora battete 3DOG e RETURN. Se questa procedura non funziona, allora avete perso il programma e dovete ricaricare il DOS.

SPECIFICA DEL VOLUME

Volume è un altro parametro che potete inserire nei comandi DOS eccetto che in CATALOG. Questo parametro controlla che il dischetto inserito nel drive sia quello giusto.

CATALOG ignora il parametro di volume perchè lo visualizza sempre nella lista di catalogo!

Per inserire il parametro di volume ponete una virgola e poi battete V seguito dal numero di volume. Ecco un esempio:

```
LOAD ZONE,V191
```

Se il numero di volume specificato non è quello letto sul dischetto vi viene segnalato l'errore VOLUME MISMATCH.

I numeri di volume devono essere compresi tra 1 e 254. Se non lo indicate o gli date il valore 0, il parametro di volume viene ignorato.

Potete usare il parametro di volume assieme a quelli di drive e di slot. Ecco alcuni esempi:

```
LOAD CARGO,D2,V24
RUN PAYROLL,S6,V111
```

ALTRI COMANDI PER IL DISCO II

Finora abbiamo illustrato il comando CATALOG, per vedere quali file sono su un dischetto, e i comandi LOAD e RUN per caricare ed eseguire un programma.

Vediamo ora i comandi, per scrivere e leggere un nuovo file, INIT, SAVE, DELETE, LOCK, RENAME e VERIFY.

INIT

Prima di potere scrivere su un dischetto ricordatevi che dovete sempre *inizializzarlo*. Quando un dischetto viene inizializzato, tutti i file che conteneva vengono cancellati. Fate quindi attenzione a non inizializzare un dischetto che contiene dei file che volete conservare.

Il comando INIT memorizza sul dischetto il programma che in quel momento è contenuto nella memoria del calcolatore. Per questo motivo tale programma prende il nome di *programma iniziale* e viene sempre eseguito ogni volta che fate il booting di questo dischetto. Il programma iniziale può essere semplicissimo oppure molto lungo. Supponete, per esempio, di avere un file di indirizzi su un dischetto.

Potete allora usare il programma di gestione degli indirizzi come programma iniziale. Quando farete, allora, il booting del dischetto il programma degli indirizzi sarà caricato e subito eseguito. Un esempio invece di programma molto semplice è quello costituito dalle due sole istruzioni NEW ed END. Ogni volta che fate il booting ottenete subito il carattere di pronto del vostro BASIC (> o]).

Un buon programma iniziale serve per dirvi qualcosa sul contenuto del dischetto.

Un esempio tipico di programma iniziale è il seguente:

```
100  TEXT
200  CALL  - 936
300  PRINT "QUESTO E' IL MIO PRIMO DISCHETTO"
400  PRINT
500  PRINT "CHE HO INZIALIZZATO IL 23/4/82"
600  PRINT
700  PRINT "SU UN SISTEMA CON 48K DI MEMORIA"
710  PRINT "E CON IL DOS VERSIONE 3.3"
800  END
```

Il nome, del programma iniziale, deve essere indicato quando usate il comando INIT. Dovete sempre sincerarvi però che esista sul dischetto questo nome. Se per caso cancellate il programma iniziale (vedremo dopo come farlo), commetterete l'errore FILE NOT FOUND ogni volta che farete il booting di questo dischetto. L'unico modo per evitare questo tipo di errore è quello di porre sul dischetto un programma iniziale e ricordare il suo nome. Per ricordare qual è il programma iniziale di un dischetto, potete prenderne nota separatamente oppure usare *sempre* lo stesso nome. Normalmente noi usiamo il nome HELLO.

Uso del comando INIT

Un esempio di comando INIT è il seguente:

```
INIT HELLO,S6,D1,V36
```

Come potete immaginare i parametri di drive, slot e volume sono opzionali. Se indicate il parametro di volume, il DOS *assegna* tale numero di volume al dischetto. INIT è l'unico comando per assegnare il numero di volume. Se non ponete il numero di volume, il DOS assegna il valore implicito 254.

Se non indicate il numero di drive o di slot, il DOS usa il valore precedentemente specificato. Per questo motivo se avete dei dubbi ponete voi esplicitamente questi due parametri.

Per inizializzare un nuovo dischetto, inseritelo nel drive al posto del System Master Disckette. Date il comando NEW per cancellare la memoria centrale. Battete quindi un programma iniziale e, per precauzione, provate ad eseguirlo prima di utilizzarlo.

Assegnate un numero di volume; per esempio 123 e battete:

```
INIT HELLO,S6,D1,V123
```

Controllate che il portellino del drive sia chiuso e battete RETURN. La spia rossa del drive si deve accendere accompagnata dal solito ronzio. Il processo di inizializzazione dura circa due minuti. Alla fine provate a dare il comando CATALOG per vedere che cosa c'è sul dischetto. Il risultato dovrebbe essere:

```
DISK VOLUME 123

 I 002 HELLO
```

Se invece usate l'Applesoft dovreste vedere:

```
DISK VOLUME 123

 A 002 HELLO
```

La lettera I significa che il programma iniziale è stato scritto in Integer BASIC, mentre la A si riferisce all'Applesoft. Il numero 002 significa che il programma è lungo 2 settori. Questo numero ovviamente sarà maggiore per programmi più lunghi.

Vi consigliamo ora di preparare una etichetta su cui porre tutte le informazioni utili di questo dischetto: numero di volume, nome del programma iniziale, ecc. ed attaccare tale etichetta sul dischetto. Se volete potete ora ripetere il booting mediante il vostro nuovo dischetto.

SAVE

Se avete seguito passo passo tutti gli esempi sin qui fatti, ora avrete probabilmente un dischetto appena inizializzato nel drive 1 collegato allo slot 6 e in memoria sarà ancora presente il programma iniziale. Provate a dare il comando LIST per leggere questo programma iniziale; se manca battete allora:

```
LOAD HELLO
```

Il comando SAVE trasferisce i programmi sul dischetto. Per *salvare* un'altra copia del programma iniziale battete il comando SAVE seguito dal nome del programma. In questo caso diamo al programma il nome PROGRAMMA INIZIALE, ma potremmo dargli qualunque altro nome; battiamo allora:

```
SAVE PROGRAMMA INIZIALE
```

A conferma della registrazione udirete il solito ronzio provenire dal disco e poi alla fine appariranno sullo schermo il carattere di pronto e il cursore. Provate ora a vedere il contenuto del catalogo che dovrebbe essere così:

```
DISK VOLUME 123

 A 002 HELLO
 A 002 PROGRAMMA INIZIALE
```

In questo modo potete salvare qualunque programma.

Se in questa fase avete usato un nome di programma che è già contenuto nel dischetto, allora il nuovo programma sostituirà completamente il vecchio che porta lo stesso nome. In questo modo il vecchio programma viene automaticamente cancellato. Attenzione però che questa sostituzione avviene solo se i due programmi sono stati scritti nello stesso tipo di BASIC. Diversamente il nuovo programma non viene accettato e sullo schermo appare il messaggio di errore FILE TYPE MISMATCH.

DELETE

Il comando DELETE permette di cancellare i programmi dal dischetto.

Se per esempio volete cancellare il programma iniziale, che avete appena registrato, potete dare uno dei seguenti comandi a seconda di quali parametri vi interessa rendere espliciti:

```
DELETE PROGRAMMA INIZIALE,S6,D1,V123
DELETE PROGRAMMA INIZIALE,V123
DELETE PROGRAMMA INIZIALE
```

Ricordatevi che potete usare i parametri di drive, di slot e di volume nell'ordine che preferite, oppure non indicarne nessuno se il dischetto è nel drive che in quel momento è considerato default.

LOCK

Se alcuni file sono così importanti da non dover mai essere cancellati, allora potete usare uno speciale comando DOS per proteggerli, cioè per impedire che un vostro comando errato li cancelli. Il comando LOCK serve appunto per *proteggere (lock)* un file. Battete LOCK seguito dal nome del file ed eventualmente dai parametri opzionali di drive, di slot e di volume. Ecco un esempio:

```
LOCK HELLO
```

È buona norma proteggere sempre il programma iniziale di un dischetto.

Se tentate di cancellare (DELETE) un file protetto, vi viene segnalato l'errore FILE LOCKED.

Se tentate di salvare un programma, che porta lo stesso nome di un programma

già salvato e protetto, ma scritto in un diverso BASIC, allora ottenete l'errore FILE TYPE MISMATCH.

I file protetti sono contrassegnati con un asterisco (*) nel catalogo di un dischetto.

UNLOCK

Per togliere la protezione ad un file protetto dovete usare il comando UNLOCK. Dopodichè potete cancellare o scrivere sopra a tale file. Per esempio potete battere:

```
UNLOCK HELLO
```

Anche in questo caso potete aggiungere al comando i parametri di drive, di slot e di volume e porli con qualunque ordine.

RENAME

Se lo ritenete necessario, potete sempre cambiare i nomi dei file su un dischetto. Un metodo poco pratico, ma possibile, consiste nel caricare in memoria il file, cancellarlo dal disco e poi riregistrarlo con il nuovo nome.

Il comando RENAME permette invece di cambiare direttamente il nome di qualunque file sul dischetto (sia file in BASIC, che file di testi, che file binari) in questo modo:

```
RENAME OLDNAME,NEWNAME
```

Attenzione però che il DOS *non* controlla se il nuovo nome è già presente sul dischetto, per cui potreste avere due file con lo stesso nome, se esso era già usato sul dischetto. Questo inconveniente può crearvi molta confusione e portarvi ad errori ben difficilmente correggibili.

Non cambiate mai il nome del programma iniziale fin tanto che non ponete un nuovo programma con lo stesso nome del vecchio programma iniziale. Il comando RENAME non cambia infatti il nome del programma a cui fa riferimento il DOS, come programma iniziale, quando esegue il booting.

Non potete cambiare il nome di un file protetto.

Come per gli altri comandi DOS, anche in questo caso potete usare i parametri di drive, di slot e di volume.

VERIFY

Il comando VERIFY vi permette di controllare se un file su dischetto è integro; cioè se, per qualunque motivo, una parte di esso è stata rovinata. Il formato del comando VERIFY è il seguente:

```
VERIFY HELLO,V123
```

Per comprendere come opera questo comando, dovete prima sapere che quando un file viene registrato, accanto ad ogni settore viene posto un numero detto *checksum.* Tale valore di checksum è calcolato in funzione del contenuto del settore stesso e registrato anche lui sul disco. Quando date il comando VERIFY, il DOS fa ricalcolare tutti i valori di checksum del file e li confronta con quelli già registrati in precedenza. È ovvio che se un settore è stato alterato, vi sarà discordanza tra il vecchio e il nuovo valore di checksum. In questo caso il calcolatore vi segnala l'errore I/O ERROR.

Se tutti i valori di checksum corrispondono a quelli registrati, il calcolatore non vi segnala niente e alla fine appare il carattere di pronto del BASIC e il cursore.

Come per gli altri comandi DOS anche in questo caso potete usare i parametri di drive, di slot e di volume.

USO DEI COMANDI DOS NEI PROGRAMMI

Nei paragrafi precedenti abbiamo descritto i comandi DOS, sottintendendo sempre che essi fossero dati in modo immediato dalla tastiera. È possibile però usarli anche come istruzioni di un programma BASIC, così da permettere una gestione dei file direttamente entro un programma.

I comandi DOS, per essere usati in un programma, devono fare parte di un stringa, posta in una istruzione PRINT, ed essere preceduti dal carattere CTRL-D (codice ASCII 4). Questa stringa di comando deve essere la prima nella lista della PRINT e se esiste una PRINT precedente, questa non deve terminare con una virgola o con un punto e virgola.

Siccome però il carattere CTRL-D non è stampabile, per evidenziarlo nel programma, che state scrivendo, non potete fare altro che porre un commento REM che documenti la sua presenza. Ecco un esempio:

```
1000 PRINT "RUN MENU" : REM C'E' UN
     CTRL-D TRA LE PRIME VIRGOLETTE
     E LA R DI RUN
```

Se volete potete in precedenza definire una variabile come CTRL-D e poi usare questa come carattere che precede i comandi DOS. Nei nostri programmi usiamo normalmente per questo scopo la variabile D$, ma voi potete usare la variabile che preferite. Il vantaggio di usare sempre la stessa variabile è quello di rendere i programmi intercambiabili. Aggiungete allora questa linea ai vostri programmi:

```
10 D$ = "": REM C'E' CTRL-D TRA LE
   VIRGOLETTE
```

se lavorate in Applesoft potete anche scrivere:

```
10 D$ =  CHR$ (4): REM  CHR$(4) = CTRL-D
```

Provate allora ad eseguire questo programma:

```
10 D$ = "": REM  CTRL-D
20  FOR I = 1 TO 10
30  PRINT D$;"CATALOG"
40  NEXT I
50  END
```

Se avete sino a 18 file sul dischetto, vedrete visualizzare il catalogo dieci volte; se ne avete di più dovrete premere un tasto qualunque ogni volta che la visualizzazione si ferma.

USO DEI FILE SU DISCO

Con il DOS dei calcolatori Apple potete usare due tipi di file su disco: i *file sequenziali* e i *file ad accesso diretto.* Ambedue questi tipi di file contengono dei blocchi di dati chiamati *campi.* Un campo può essere lungo un solo carattere oppure molti caratteri.

File sequenziali

I file sequenziali, come dice lo stesso nome, possono essere usati solo in maniera sequenziale. Ciò significa che per leggere (o scrivere) l'ultimo campo del file, dovete prima aver letto (o scritto) tutti i campi precedenti. In alcuni casi applicativi i file sequenziali sono preferibili ai file ad accesso diretto.

File ad accesso diretto

I file ad accesso diretto sono più flessibili di quelli sequenziali. Potete infatti leggere o scrivere un campo direttamente in qualunque posizione del file si trovi. Molto spesso questi file sono i più usati.

USO DEI FILE SEQUENZIALI

Prima di poter usare i file sequenziali dovete imparare altri comandi DOS: OPEN, CLOSE, READ e WRITE.

Apertura dei file sequenziali

Prima di accedere ad un file bisogna aprirlo. Aprire un file significa che il DOS ricerca delle informazioni relative a questo file: se esso è già sul disco, e se lo è già dove si trova esattamente. Il comando OPEN fa predisporre anche una certa area di memoria che verrà usata come *buffer* (cioè come *memoria tampone*). Il buffer permette di accedere solo saltuariamente al disco e non ogni volta che il programma ha un singolo dato da leggere o da scrivere. In altre parole l'accesso al disco avviene per blocchi (o settori) e non per singolo dato con un gran risparmio di tempo. Un esempio di comando OPEN è il seguente:

```
OPEN FILENAME,S6,D2,V99
```

Se il file non esiste già allora il DOS ne crea uno nuovo inserendo il suo nome nella directory del dischetto.

Il comando OPEN, inserito in un programma, deve comparire in una PRINT preceduto dal carattere CTRL-D.

Il programma seguente apre il file SEQUENZIALE sul dischetto posto nel drive default:

```
100 D$ = "": REM CTRL-D
200 PRINT D$;"OPEN SEQUENZIALE"
300 END
```

Chiameremo questo programma *Programma 1.*

Nel comando OPEN possiamo inserire i soliti parametri di drive, di slot e di volume:

```
100 D$ = "": REM CTRL-D
200 PRINT D$;"OPEN SEQUENZIALE,S6,D1,V123"
300 END
```

oppure:

```
200 PRINT D$;"OPEN SEQUENZIALE,V123,S6"
```

Notate che i parametri sono posti prima della chiusura delle virgolette e sono separati da virgole.

Dopo aver eseguito questo programma vedrete che il nuovo file SEQUENZIALE è presente nel catalogo del dischetto. Provate infatti a dare il comando CATALOG e dovreste vedere sullo schermo qualcosa del genere:

```
DISK VOLUME 123

*A 002 HELLO
 T 001 SEQUENZIALE
```

La T che compare davanti al nome SEQUENZIALE sta ad indicare che si tratta di un file di testo. L'asterisco davanti al file HELLO indica invece che quest'ultimo è stato protetto.

Chiusura dei file

Un programma che apre un file deve sempre anche chiuderlo. Il nostro esempio, Programma 1, presenta infatti la grave limitazione di non chiudere il file SEQUENZIALE che ha aperto. La non chiusura di un file può portare alla perdita di dati come vi abbiamo già illustrato parlando dei dischi rovinati nel software. Il comando CLOSE può essere dato con due formati. Il primo è il seguente:

```
CLOSE
```

Il comando CLOSE senza parametri, chiude tutti i file aperti su tutti i dischetti posti in qualunque drive.

Se volete invece chiudere uno specifico file, allora dovete indicare il suo nome nella lista di CLOSE:

```
CLOSE FILENAME
```

In ambedue i casi non dovete mai indicare i parametri di drive, di slot o di volume perchè il DOS sa già dov'è il file che in precedenza aveva aperto.

Il Programma 1 deve quindi essere corretto aggiungendo la linea 290:

```
290  PRINT D$;"CLOSE"
```

oppure:

```
290 PRINT D$;"CLOSE SEQUENZIALE"
```

Scrittura in un file sequenziale

Iniziamo a vedere come si deve procedere per scrivere in un file sequenziale.

Per prima cosa dobbiamo precisare che le informazioni sono inviate al disco nello stesso modo come sarebbero inviate allo schermo o alla stampante mediante l'istruzione PRINT. Infatti ogni cosa che stampate la potete anche porre in un file su disco. Potete cosi immaginare un file sequenziale come se fosse uno schermo o, ancora meglio, come la carta di un stampante.

Quando scrivete qualcosa in un file, il DOS aggiorna un puntatore interno che punta alla prossima posizione sul disco dove i dati saranno registrati, cosi come la stampante fa avanzare la carta di una riga.

Il puntatore di un file sequenziale può solo muoversi in avanti. Il comando OPEN lo fa invece tornare all'inizio.

Prima di scrivere dei dati in un file è necessario però avvisare il DOS, mediante un comando WRITE, che l'istruzione PRINT servirà per portare i dati sul disco e non sullo schermo. Nel caso dei file sequenziali l'istruzione WRITE si presenta così:

```
WRITE FILENAME
```

Dopo che avete dato il comando WRITE, le uscite successive saranno indirizzate verso il file. In tal caso però anche i messaggi di errore andranno sul file. Dopo che un messaggio di errore sia stato registrato nel file, il comando WRITE sarà però automaticamente annullato e vedrete apparire sullo schermo il carattere di pronto e il cursore.

Il comando WRITE deve essere posto in una PRINT preceduto dal carattere CTRL-D. Se per sbaglio date il comando WRITE in modo immediato, vi viene segnalato l'errore NOT DIRECT COMMAND.

Aggiungete quindi le seguenti linee al Programma 1:

```
210 PRINT D$;"WRITE SEQUENZIALE"
220 PRINT "IL TESTO VIENE SCRITTO NEL FILE"
```

Un programma può allora fare le seguenti cose:

1. Creare un file.
2. Aprire un file.
3. Memorizzare un testo nel file.
4. Chiudere il file.

Potete inserire quante istruzioni PRINT volete tra la linea 210 e quella 290. In altre parole potete scrivere nel file sia un testo, che dei numeri, che delle variabili. Per esempio:

```
220  FOR I = 1 TO 100
230  PRINT I
240  NEXT I
250  PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

Notate che il carattere CTRL-D è usato solo davanti ai comandi DOS e non davanti al contenuto che viene scritto nel file.

Attenzione: non potete usare le istruzioni FLASH e INVERSE per scrivere in un file.

Se eseguite più volte questo programma, verrà ogni volta riscritto sopra lo stesso file. Questo significa che se in una delle volte successive scrivete meno dati sul file, rimarrà presente nel file stesso la parte finale delle registrazioni precedenti. In altre parole solo se scrivete dei file eguali o sempre più lunghi, siete sicuri di cancellare il contenuto precedente.

Per ovviare a questo inconveniente è necessario cancellare il file prima di rieseguire il programma. Inserite allora una PRINT nella cui lista vi sia il comando DELETE, prima di aprire il file con OPEN. In questo modo ogni volta che eseguite il programma dapprima cancellate il vecchio file e poi lo aprite nuovamente.

Attenzione però che la prima volta che eseguite questo programma otterrete un errore perchè il comando DELETE cercherà di cancellare un file di nome SEQUENZIALE che non è mai stato posto sul dischetto. L'errore che vi verrà segnalato sarà FILE NOT FOUND.

Per evitare questo dovete porre un altro comando OPEN proprio prima di DELETE. Ecco quello che succede:

1. Il primo OPEN crea un file, se già non esiste.
2. DELETE cancella il file indipendentemente da quando è stato creato.
3. Il secondo comando OPEN crea un nuovo file vuoto.

Se fate questi cambiamenti nel Programma 1, ricordatevi che se avete specificato i parametri di drive, di slot e di volume non c'è più bisogno di indicarli fin tanto che non li cambiate. In questo caso i parametri specificati vengono detti di default (cioè impliciti).

Il Programma 1 risulta quindi il seguente:

```
100 D$ = "": REM  CTRL-D
110  PRINT D$;"OPEN SEQUENZIALE,V123,S6,D1"
120  PRINT D$;"DELETE SEQUENZIALE"
200  PRINT D$;"OPEN SEQUENZIALE"
210  PRINT D$;"WRITE SEQUENZIALE"
220  PRINT "IL TESTO VIENE SCRITTO NEL FILE"
290  PRINT D$;"CLOSE"
300  END
```

Ogni volta che eseguite questo programma esso memorizzerà nel file SEQUENZIALE ciò che è riportato nelle istruzioni PRINT comprese tra le linee 210 e 290.

Il nome del file non è necessario che sia definito quando scrivete il programma. Potete benissimo usare una variabile e fornire il nome come dato d'ingresso. Ecco un esempio:

```
10  INPUT "NOME FILE: ";F$
100 D$ = "": REM  CTRL-D
110  PRINT D$;"OPEN ";F$;",V123,S6,D1"
120  PRINT D$;"DELETE ";F$
```

```
200  PRINT D$;"OPEN F$"
210  PRINT D$;"WRITE ";F$
220  PRINT "IL TESTO VIENE SCRITTO NEL FILE"
290  PRINT D$;"CLOSE"
300  END
```

Oltre al nome del file il programma può anche chiedervi i valori dei parametri di drive, di slot e di volume:

```
10  INPUT "NOME FILE: ";F$
20  INPUT "NUMERO SLOT: ";S
30  INPUT "NUMERO DRIVE: ";D
40  INPUT "NUMERO VOLUME: ";V
100 D$ = "": REM  CTRL-D
110  PRINT D$;"OPEN ";F$;",S";S;",D";D;",V";V
```

Ponete molta attenzione alla sintassi di queste istruzioni e in modo particolare alle virgole davanti a S, D e V. Se le dimenticate il DOS non è più in grado di distinguere i parametri dal nome del file.

Per far sì che questo programma sia veramente utile, è necessario però che voi possiate dare il testo del file come dato d'ingresso. Per questo motivo dovete usare almeno una istruzione INPUT come in questo esempio:

```
150  INPUT "DARE IL TESTO DEL FILE: ";T$
220  PRINT T$
```

In questo caso però potete dare un ingresso solo una linea di testo. Se voleste dare più linee dovreste inserire più istruzioni INPUT. È preferibile invece usare una sola INPUT, ma ripeterla più volte dopo aver verificato una condizione di controllo. Nell'esempio che segue si vede che con la GOTO della linea 230 si ripete la INPUT; se poi viene data la parola END il programma salta alla chiusura del file:

```
150  INPUT "DARE IL TESTO DEL FILE: ";T$
160  IF T$ = "END" THEN 290
210  PRINT D$;"WRITE ";F$
220  PRINT T$
230  GOTO 150
```

Potete allora battere un testo lungo quanto volete ed alla fine terminarlo con la parola END (FINE).

Esiste però ancora un problema. Il messaggio di avviso della INPUT, "DARE IL TESTO DEL FILE", non apparirà sullo schermo, ma andrà anche lui sul file dopo l'esecuzione del comando WRITE (vi ricordiamo che dopo WRITE tutte le uscite vanno sul file).

È necessario quindi cancellare il comando WRITE quando non si vuole mandare una uscita sul file. Qualunque comando DOS annulla WRITE, ma il metodo più sicuro consiste nell'usare il *comando nullo.* Il comando nullo è dato dal solo carattere CTRL-D. Aggiungete quindi questa linea al vostro programma:

```
225  PRINT D$
```

In conclusione questo programma vi permette di scrivere in un file un testo lungo quanto volete (nei limiti della capacità del dischetto), di dare al file il nome che volete e di usare qualunque drive connesso al vostro calcolatore Apple II.

Lettura dei file sequenziali

La lettura di un file sequenziale avviene nello stesso modo di come si effettua la scrittura salvo che si deve dare il comando READ invece di WRITE. Il comando READ deve essere seguito dal nome del file:

```
READ FILENAME
```

Il comando READ deve essere posto in una istruzione PRINT preceduto dal carattere CTRL-D. Se date il comando READ in modo immediato ottenete l'errore NOT DIRECT COMMAND.

Dopo l'esecuzione di READ, le successive istruzioni INPUT ricevono i dati dal file sul dischetto sin tanto che un altro comando DOS, o un errore, cancellano il comando READ.

Il *Programma 2* seguente mostra l'uso del comando READ:

```
100 D$ = "": REM  CTRL-D
110  INPUT "FILE NAME TO READ:  ";F$
120  INPUT "SLOT NUMBER:  ";S
130  INPUT "DRIVE NUMBER:  ";D
140  INPUT "VOLUME NUMBER:  ";V
150  PRINT D$;"OPEN ";F$;",S";S;",D";D;",V";V
160  PRINT D$;"READ ";F$
170  INPUT A$
180  PRINT A$
190  GOTO 170
200  END
```

Il Programma 2 visualizza il file creato dal Programma 1. Al termine del file vi appare il messaggio END OF DATA e il programma si ferma con l'avviso BREAK IN 170.

Potete notare che nel Programma 2 manca il comando CLOSE. Vi abbiamo già spiegato infatti che il comando CLOSE è importantissimo solo quando si scrive in un

file o per aggiornare la sua directory. Nel caso invece della lettura, la chiusura del file è raccomandabile solo per motivi di sicurezza.

Un'altra considerazione deve essere fatta a riguardo del messaggio finale che viene visualizzato assieme al testo del file, cioè al fatto che il file è stato letto sino al raggiungimento di una condizione di errore.

Come prevenire l'errore END OF DATA

La condizione END OF DATA può essere rilevata prima che sia generato uno stato di errore. Il Programma 2 può saltare alla chiusura del file dopo aver rilevato la fine del file. Il modo più semplice per saltare, in caso di errore, è quello di usare l'istruzione dell'Applesoft ONERR GOTO (vedere il capitolo 4) che però non è disponibile in Integer BASIC. (Per i codici di errore del DOS vedere l'Appendice C).

Un altro metodo che può funzionare sia in Applesoft che in Integer BASIC è il seguente. Modificate il programma 1 in modo da scrivere nel file, come ultimo carattere, un carattere speciale che voi avete stabilito sia il segno di fine del file. Per esempio questo carattere lo potete far scrivere dopo che avete dato in ingresso la parola finale END. Modificate poi anche il Programma 2 in modo che esso controlli la presenza di questo carattere di fine del file. Quando il carattere di fine del file viene incontrato, il Programma 2 chiude correttamente il file.

Differenze tra Integer BASIC e Applesoft

Il comando READ identifica un file su disco come sorgente di dati per le successive istruzioni INPUT. Tali istruzioni INPUT devono però essere conformi al tipo di BASIC con cui state scrivendo il vostro programma (vedere il capitolo 8). In ogni caso i dati letti dal file dipendono sempre da come sono stati scritti in precedenza nel file stesso.

Per esempio in Applesoft:

```
100 D$ = "": REM  CTRL-D
200  PRINT D$;"OPEN FILE1"
300  PRINT D$;"WRITE FILE1"
400  PRINT "HELLO","QUESTO E' IL TESTO"
500  PRINT D$;"CLOSE"
600  END
```

oppure con questa piccola modifica:

```
400  PRINT "HELLO","QUESTO E' IL TESTO"
```

vengono memorizzate *due* linee di testo nel FILE 1.

L'Applesoft non tratta le virgole, come separatori, nello stesso modo di come le tratta l'Integer BASIC. Se provate infatti a leggere il FILE 1 con le istruzioni:

```
300  PRINT D$;"READ FILE1"
400  INPUT A$
```

in Applesoft vi viene segnalato l'errore? EXTRA IGNORED e la variabile A$ víene posta eguale a HELLO, indipendentemente da quale istruzione 400 avete usato. In Applesoft le virgole separano gli elementi di una stessa PRINT.

In Integer BASIC invece ambedue le linee 400 vengono accettate come una unica linea; così nel primo caso A$ è uguale a HELLO QUESTO E' IL TESTO, mentre nel secondo è uguale a HELLO, QUESTO E' IL TESTO.

Uso dell'istruzione GET in Applesoft per leggere il testo di un file

In Applesoft è possibile usare l'istruzione GET per leggere i dati di un file invece dell'istruzione INPUT. La differenza consiste nel fatto che GET legge *un carattere alla volta.* Così se un file contiene il testo: QUESTO E' IL TESTO DI UN FILE e voi accedete al file mediante le istruzioni OPEN, READ, e GET, il primo carattere che leggete è Q. Successivamente GET legge la lettera U, poi la E e così avanti. Se la GET legge una virgola o un ritorno del carrello, il programma può rilevarli e prendere le giuste decisioni.

Il Programma 3 seguente, valido solo per l'Applesoft, effettua la lettura di un file mediante l'istruzione GET, ricomponendo le singole linee del testo:

```
100 D$ =  CHR$ (4): REM  CTRL-D
200  INPUT "FILE NAME TO READ: ";F$
300  INPUT "SLOT NUMBER: ";S
400  INPUT "DRIVE NUMBER: ";D
500  INPUT "VOLUME NUMBER: ";V
600  PRINT D$;"OPEN ";F$;",V";V;",D";D;",S";S
700  PRINT D$;"READ ";F$
800 B$ = ""
900  GET A$
1000  IF A$ =  CHR$ (13) THEN 1300
1100 B$ = B$ + A$
1200  GOTO 900
1300  REM  RETURN CHARACTER FOUND
1400  REM  B$ IS COMPLETE
1500  PRINT B$
1600  GOTO 800
1700  END
```

Attenzione però che l'istruzione GET presenta un problema quando è usata con i file su disco. Infatti il primo carattere stampato da una PRINT, dopo che una GET sia stata eseguita, viene sempre trascurato. Di conseguenza se questo primo carattere, nella lista di una PRINT, è CTRL-D il comando DOS non viene inteso come tale, ma viene semplicemente visualizzato.

Per ovviare a questo inconveniente bisogna inserire, come primo carattere della

PRINT, un carattere che possa *essere perso.* Un buon carattere per questo scopo è CRTL-A (codice ASCII 1) che non è stampabile e non ha uno speciale significato come CTRL-D.

Il Programma 3 può essere allora corretto in questo modo alla linea 1500:

```
1500  PRINT  CHR$ (1);B$: REM  CHR$(1) = CTRL-A
```

Registrazione di numeri in un file

Forse avete già registrato dei numeri in un file come parte di un testo:

```
220 PRINT "IL MIO INDIRIZZO E' VIA PASCAL 225"
```

oppure direttamente proprio come valori numerici:

```
230 PRINT 1,2,3,4,5
```

oppure mediante variabili numeriche:

```
240  PRINT A,B,C,D,E
```

Se ora usate il Programma 2 per rileggere i numeri, registrati direttamente, noterete uno strano risultato.

In Integer BASIC i numeri separati da virgole o punti e virgola saranno letti come un unico grande numero. Per esempio i numeri 1, 2, 3, 4 e 5, dell'esempio precedente, appaiono come un unico numero 12345.

In Applesoft le cose sono molto più complicate. I primi tre numeri (123) si presentano concatenati e stampati come una unica linea. I due valori successivi (4 e 5) producono due linee e sono stampati separatamente. In totale si hanno così tre linee in uscita.

Questi problemi nascono a causa del tipo di formato usato per memorizzare i numeri nel file. Infatti le virgole non vengono memorizzate sul disco in analogia alla loro non visualizzazione sullo schermo. Le virgole separatrici fanno apparire sullo schermo i valori nelle diverse posizioni di tabulazione, ma non appaiono esse stesse. Quando l'uscita è diretta al disco, il DOS trascura completamente le virgole. Di conseguenza sul disco i valori numerici sono tutti concatenati sin quando un carattere di ritorno del carrello li separe (codice ASCII 13). Il carattere di ritorno del carrello è l'unico carattere che il DOS interpreta come elemento separatore.

L'Integer BASIC produce un ritorno del carrello dopo la quinta virgola di tabulazione; l'Applesoft lo produce invece dopo tre punti di tabulazione.

In definitiva per evitare qualunque problema, dovete separare i campi numerici con i caratteri di ritorno del carrello quando operate in un file su disco. Il modo più

semplice per fare questo è quello di scrivere su disco ogni numero con una diversa istruzione PRINT:

```
230  PRINT 1: PRINT 2: PRINT 3: PRINT 4:
 PRINT 5
```

Un altro metodo, che potete usare in Applesoft, è quello di porre, in una stessa PRINT, CHR$ (13) come elemento separatore dei valori numerici:

```
230  PRINT 1;CHR$(13);2;CHR$(13);3;CHR$(1
 3);4;CHR$(13);5
```

In quest'ultimo caso è preferibile porre una variabile eguale a CHR$(13) e poi usare questa variabile come separatore nelle PRINT:

```
11 R$ = CHR$(13): REM CARATTERE RETURN
  .
  .
  .
230 PRINT 1;R$;2;R$;3;R$;4;R$;5
```

COME AMPLIARE UN FILE SEQUENZIALE

Quando terminate di scrivere un file sequenziale lo dovete chiudere e per questo motivo perdete conoscenza di dove l'ultimo campo è stato memorizzato. Se volete quindi aggiungere altri campi al vostro file dovete prima *trovare* la sua coda e poi scrivere i nuovi campi. Questa procedura può essere ovviamente molto laboriosa per cui è stato creato un apposito comando DOS per aggiungere altri dati ad un file sequenziale: il comando APPEND.

Il comando APPEND porta il puntatore del file alla prima posizione dopo la sua fine. Se tentate di leggere il file, dopo aver dato APPEND, ottenete l'errore END OF DATA. Se invece scrivete, i nuovi dati vengono aggiunti in coda.

APPEND non richiede che venga in precedenza aperto il file con OPEN. Esistono però due importanti differenze tra APPEND e OPEN:

1. Il comando APPEND richiede che il file esista già; APPEND non crea file nuovi. Se il file non esiste già vi viene segnalato l'errore FILE NOT FOUND.
2. APPEND porta il puntatore in *coda* al file; OPEN lo porta invece al suo *inizio.*

Il formato di APPEND è lo stesso di OPEN. Ecco un esempio:

```
APPEND FILENAME,S6,D2,V99
```

Come sempre i parametri di drive, di slot e di volume sono opzionali.

IL COMANDO DI POSIZIONE

Un altro comando DOS molto utile è POSITION che permette di muovere il puntatore del file di un certo numero di posizioni in avanti. POSITION sposta infatti il puntatore, *solo in avanti,* di un numero R di campi relativi. Ecco un esempio:

```
POSITION FILENAME,R30
```

R indica il numero relativo di campi che vengono saltati. Siccome i campi di un file sono separati dal carattere di ritorno del carrello, si può dire che il comando POSITION fa spostare il puntatore di R caratteri di ritorno del carrello. Se viene posto il parametro RO il puntatore non si sposta.

In effetti il comando POSITION esamina il file carattere per carattere per individuare quelli di ritorno del carrello. Se non vi è un numero sufficiente di campi, o viene incontrato un carattere anomalo, viene segnalato l'errore END OF DATA (anche senza avere eseguito una INPUT o una GET).

Per poter usare il comando POSITION è necessario che il file sia aperto. In effetti l'apertura del file predispone il puntatore all'inizio. Se in questo caso date il comando POSITION il parametro R individua i campi *assoluti* del file.

Ricordatevi che come ogni altro comando DOS, anche POSITION annulla i comandi READ e WRITE. Fate quindi attenzione ad usare position prima di READ e WRITE e non dopo.

USO DEI FILE AD ACCESSO DIRETTO

I file ad accesso diretto sono strutturati in sezioni chiamate *record.* Ogni record di un file contiene sempre lo stesso quantitativo di informazione; cioè uno stesso numero di byte (o caratteri). Tale numero di caratteri viene chiamato *lunghezza* del record e viene stabilito al momento della creazione del file.

I record sono identificati da un numero che rappresenta la loro posizione assoluta nel file. Il primo record ha la posizione zero; gli altri in ordine crescente hanno la posizione uno, due, ecc.

Il più corto file ad accesso diretto ha un solo record. Un file può essere esteso aggiungendo nuovi record, ma non può essere mai accorciato. Per rimuovere record non voluti in un file, o per accorciarlo, è necessario ricopiare in un nuovo file i record che si desidera mantenere.

I programmi devono specificare sia il record sia quale parte di un record, di un file ad accesso diretto, desiderano usare.

Apertura di un file ad accesso diretto

Per definire un file ad accesso diretto è sufficiente indicare un nuovo parametro L, la *lunghezza* del record, nella istruzione di apertura del file. Ecco un esempio:

```
OPEN FILENAME,L10,S6,D1,V100
```

Il parametro L, lunghezza del record, può avere un qualunque valore intero compreso tra 1 e 32767. Esso può non essere il primo parametro della lista, ma deve essere presente per definire un file ad accesso diretto.

I programmi non possono scrivere record più lunghi del valore L specificato in apertura. L comprende anche i caratteri virgola e ritorno del carrello. Se per errore un record è troppo lungo, allora esso si sovrappone al successivo creando una notevole confusione.

Chiusura di un file ad accesso diretto

Anche i file ad accesso diretto vengono chiusi con CLOSE come quelli sequenziali.

Lettura e scrittura di un file ad accesso diretto

I comandi READ e WRITE richiedono che sia specificato il *numero* R *di record* su cui operare. Il parametro R fa spostare il puntatore all'inizio del record interessato. Ecco due esempi:

```
READ FILENAME,R13
```

e:

```
WRITE FILENAME,R6
```

Oltre al parametro R possono essere presenti anche i parametri di drive, di slot e di volume. Se R è mancante il puntatore non viene spostato.

UN ESEMPIO DI FILE AD ACCESSO DIRETTO

I programmi seguenti mostrano un esempio di file ad accesso diretto. Essi possono essere scritti sia in Integer BASIC che in Applesoft. Se il secondo programma viene scritto in Integer BASIC è necessario aggiungere una istruzione DIM, prima della linea 100, per dimensionare le variabili B$ e C$ per farle contenere 255 caratteri ognuna.

Il primo programma crea un file di nome RANDOM:

```
 10  REM      RANDOM-ACCESS FILE CREATING PROGRAM
 20  REM
 30  REM      BE SURE TO RUN THIS PROGRAM FIRST AS IT
 40  REM  STORES INFORMATION IN RECORD ZERO WHICH MUST
 50  REM  EXIST FOR THE NEXT PROGRAM TO FUNCTION.
 60  REM
100  D$="" : REM  CTRL-D
200  PRINT D$;"OPEN RANDOM,S6,D1,L256" : REM  OPEN THE FILE
300  PRINT D$;"WRITE RANDOM,R0" : REM  WRITE, RECORD ZERO
400  PRINT 0 : REM  STORE A ZERO IN RECORD ZERO
500  PRINT D$;"CLOSE" : REM  CLOSE THE FILE
600  END
```

Il secondo programma permette di leggere, cambiare, aggiungere e listare i record del file. Ogni record contiene una linea di informazioni che potete battere in ingresso alla tastiera:

```
10   REM      RANDOM-ACCESS FILE DEMONSTRATION PROGRAM
20   REM
30   REM      THIS PROGRAM WILL MAINTAIN A RANDOM-ACCESS
40   REM  FILE WHICH CONSISTS OF SINGLE LINE RECORDS.
50   REM
60   REM      RECORD ZERO CONTAINS A NUMBER INDICATING
70   REM  THE LAST RECORD NUMBER IN USE.
80   REM
85   REM  NOTE:  FILE "RANDOM" MUST BE CREATED BEFORE
90   REM         ATTEMPTING TO USE THIS PROGRAM.
95   REM
100  D$="" : REM  CTRL-D
200  PRINT D$;"OPEN RANDOM,S6,D1,L256"
225  PRINT D$;"READ RANDOM,R0" : REM  READ RECORD ZERO
250  INPUT M : REM  M = THE LAST RECORD NUMBER IN USE
275  PRINT D$ : REM  CANCEL READ COMMAND
300  CALL -936 : REM  CLEAR SCREEN
400  PRINT "RANDOM-ACCESS FILE DEMONSTRATION"
500  PRINT : PRINT : PRINT : PRINT
600  PRINT "COMMANDS:" : PRINT
700  PRINT " 0  =  STOP"
800  PRINT " 1  =  READ A RECORD"
900  PRINT " 2  =  ADD A RECORD"
1000  PRINT " 3  =  CHANGE A RECORD"
1100  PRINT " 4  =  LIST ALL RECORDS"
1200  PRINT : PRINT
1300  PRINT "WHICH";
1400  INPUT C
1500  IF C=0 THEN 8300 : REM   BRANCH
1600  IF C=1 THEN 2200 : REM   TO
1700  IF C=2 THEN 3300 : REM   THE
1800  IF C=3 THEN 4600 : REM   SELECTED
1900  IF C=4 THEN 6700 : REM   ROUTINE
2000  GOTO 300         : REM   (OR RE-DISPLAY THE MENU)
2050  REM
2100  REM  * * * * *     READ A RECORD     * * * * *
2150  REM
2200  CALL -936 : REM  CLEAR SCREEN
2300  PRINT : PRINT "READ A RECORD" : PRINT
2400  PRINT "WHICH RECORD NUMBER (0 TO STOP)";
2500  INPUT R
2600  IF R<1 THEN 300 : REM  RETURN TO MAIN MENU
2650  IF R>M THEN 2200 : REM  RECORD DOES NOT EXIST
2700  PRINT D$;"READ RANDOM,R";R : REM PREPARE TO READ RECORD
2800  INPUT B$ : REM  READ THE DATA
2900  PRINT D$ : REM  CANCEL READ COMMAND
3000  PRINT : PRINT B$ : PRINT : REM  DISPLAY THE DATA
3100  GOTO 2400 : REM  ASK FOR ANOTHER RECORD NUMBER
3150  REM
3200  REM  * * * * *     ADD A RECORD     * * * * *
```

```
3250  REM
3300  CALL -936 : REM  CLEAR SCREEN
3400  PRINT : PRINT "ADD A RECORD" : PRINT
3500  PRINT "NEXT RECORD NUMBER = ";M+1
3600  PRINT : PRINT "ENTER DATA FOR RECORD ";M+1
3625  PRINT "(PRESS [RETURN] NOW TO STOP ADDING)"
3700  INPUT B$ : REM  GET USER'S RESPONSE
3750  IF B$ = "" THEN 300 : REM  QUIT IS JUST [RETURN]
3800  PRINT D$;"WRITE RANDOM,R";M+1 : REM  PREPARE TO WRITE
3900  PRINT B$ : REM  SEND DATA TO FILE
4000  M = M + 1 : REM  INCREMENT LAST RECORD NUMBER
4100  PRINT D$;"WRITE RANDOM,RO" : REM  PREPARE TO WRITE
  RECORD ZERO
4200  PRINT M : REM  STORE UPDATED VALUE
4300  PRINT D$ : REM  CANCEL WRITE COMMAND
4400  GOTO 3500 : REM  LOOP FOR ANOTHER RECORD
4450  REM
4500  REM  * * * * *     CHANGE A RECORD     * * * * *
4550  REM
4600  CALL -936 : REM  CLEAR SCREEN
4700  PRINT : PRINT "CHANGE A RECORD" : PRINT
4800  PRINT "CHANGE WHICH RECORD (O TO STOP)";
4900  INPUT R
5000  IF R<1 THEN 300 : REM  RETURN TO MAIN MENU
5050  IF R>M THEN 4600 : REM  TRY AGAIN IF RECORD NOT ON FILE
5100  PRINT D$;"READ RANDOM,R";R : REM  PREPARE TO READ
5200  INPUT B$ : REM  READ THE RECORD
5300  PRINT D$ : REM  CANCEL READ COMMAND
5400  PRINT : PRINT B$ : PRINT : REM  DISPLAY THE DATA
5500  PRINT "ENTER THE NEW DATA"
5600  PRINT "(PRESS [RETURN] NOW TO CANCEL CHANGES)"
5700  PRINT
5800  INPUT C$ : REM  GET USER'S RESPONSE
5900  IF C$>"" THEN 6200 : REM  BRANCH IF NEW DATA
6000  PRINT "RECORD ";R;" UNCHANGED ! ! !" : REM  LOOP IF
6100  GOTO 4800 : REM  NO CHANGES DESIRED
6200  PRINT D$;"WRITE RANDOM,R";R : REM  PREPARE TO WRITE
6300  PRINT C$ : REM  STORE CHANGED DATA
6400  PRINT D$ : REM  CANCEL WRITE COMMAND
6500  GOTO 4800 : REM  LOOP FOR ANOTHER RECORD TO CHANGE
6550  REM
6600  REM  * * * * *     LIST ALL RECORDS     * * * * *
6650  REM
6700  CALL -936 : REM  CLEAR SCREEN
6800  PRINT : PRINT "LIST ALL RECORDS" : PRINT
6900  R = 0 : REM  RESET THE COUNTER
7000  R = R + 1 : REM  INCREMENT THE COUNTER
7100  IF R > M THEN 7700 : REM  STOP AFTER LAST RECORD
7200  PRINT D$;"READ RANDOM,R";R : REM  PREPARE TO READ
7300  INPUT B$ : REM  READ THE DATA
7400  PRINT "RECORD NUMBER = ";R : REM  DISPLAY RECORD NUMBER
7500  PRINT B$ : PRINT : REM  DISPLAY RECORD'S DATA
7600  GOTO 7000 : REM  LOOP FOR NEXT RECORD
7700  PRINT D$ : REM  CANCEL READ COMMAND
```

```
7800  PRINT : PRINT "* * * * END-OF-FILE" : REM  PRINT
 EOF MESSAGE
7900  PRINT "PRESS RETURN TO CONTINUE"; : REM  REQUEST
 RESPONSE
8000  INPUT B$ : REM  GET USER'S RESPONSE
8100  GOTO 300 : REM  RETURN TO MAIN MENU
8150  REM
8200  REM  * * * * *      STOP PROGRAM      * * * * *
8250  REM
8300  PRINT D$;"CLOSE" : REM  CLOSE THE FILE
8400  CALL -936 : REM  CLEAR SCREEN
8500  FOR I=1 TO 24 : PRINT : NEXT I : REM  MOVE TO BOTTOM
 LINE
8600  PRINT "PROGRAM COMPLETE."
8700  END
```

IL PARAMETRO BYTE

Byte è un altro importante parametro dei file ad accesso diretto. Esso può essere usato con i comandi READ, WRITE e POSITION per portare il puntatore su un byte (o carattere) di un certo record.

Il formato di questo comando prevede una lettera B seguita dal numero progressivo di byte nel record. Il parametro di byte implica che specifichiate anche quello di record. Ecco un esempio:

```
READ FILENAME,R19,B3
```

In questo esempio la lettura inizierà dal quarto byte del 20-esimo record. Ricordatevi infatti che il primo record è il record 0. Il parametro di byte può muovere il puntatore sia in avanti che indietro.

Quando usate il parametro di byte dovete conoscere molto bene la distribuzione dei dati nel record; diversamente potreste ottenere dei dati completamente privi di significato. In altre parole dovete conoscere bene la posizione dei byte di ogni singolo dato o campo per potervi accedere correttamente.

Quando usate il parametro di byte nel comando POSITION, ricordatevi che POSITION cancella ogni precedente comando READ e WRITE. Potete quindi eseguire un comando READ e WRITE che includa il parametro di record, poi eseguire POSITION per muovere il puntatore al campo voluto, ma poi alla fine dovete rieseguire il comando iniziale READ o WRITE.

POSITION permette di indirizzare i campi di un record (solo in avanti). Nell'ambito del campo potete poi usare il parametro di byte per accedere ad ogni singolo carattere.

ALTRI COMANDI DOS

Descriviamo ora altri comandi del DOS: EXEC, MAXFILES, TRACE, MON e NOMON.

EXEC

Questo comando permette di passare il controllo del calcolatore al testo di un file. Ecco un esempio di comando EXEC:

```
EXEC FILENAME,R6,S5,D2,V23
```

Il parametro R è simile a quello del comando POSITION e indica il numero relativo di campo. Dal momento però che EXEC apre il file e posiziona il puntatore al suo inizio, il parametro R viene ad indicare la posizione assoluta dei campi. RO è il valore di default.

Tutti i parametri del comando EXEC sono opzionali.

Quando EXEC viene eseguito, il file indicato viene aperto e sono subito eseguite implicitamente le istruzioni READ e INPUT. La linea indicata da R è così prelevata dal file (se R manca viene letta la prima linea) e viene trattata come se fosse una linea qualunque di istruzioni appena battuta alla tastiera. Se essa è priva di significato viene segnalato l'errore? SYNTAX ERROR oppure *** SYNTAX ERR. Se invece è una linea valida come la seguente:

```
100 PRINT "QUESTO E' UN TESTO"
```

viene memorizzata come se la aveste appena battuta alla tastiera. Se rappresenta un comando valido, come LIST o RUN, allora viene subito eseguita.

Dopo che le indicazioni della prima linea sono state soddisfatte viene prelevata dal file la seconda linea ed anch'essa eseguita. Questa procedura continua sino alla fine del file e in quel momento il controllo del calcolatore ritorna alla tastiera.

Supponiamo per esempio che il file denominato BUBBA contenga le seguenti linee di testo:

```
PRINT "I HAVE CONTROL OF YOUR APPLE ! ! !"
FP
100  GR
200  FOR I=0 TO 39
300  FOR J=0 TO 39
400  COLOR=RND(O)*15
500  PLOT I,J
600  NEXT J
700  NEXT I
800  FOR I=1 TO 5000 : NEXT I
900  TEXT : CALL -936
999  END
```

```
LIST
FOR I=1 TO 5000 : NEXT I
RUN
NEW
PRINT "FINISHED.  HERE'S YOUR DISK CATALOG:"
CATALOG
```

Battete allora il comando:

```
EXEC BUBBA
```

dapprima vi appare un messaggio; poi viene richiamato l'Applesoft ed un piccolo programma che è listato ed eseguito dopo una breve pausa (ciclo FOR-NEXT da 1 a 5000). Successivamente questo programma viene cancellato dalla memoria; poi dopo un altro messaggio viene visualizzato il catalogo del dischetto selezionato.

Alcune note su EXEC

Solo un file EXEC alla volta può essere aperto.

Se un file richiamato da EXEC contiene un altro comando EXEC, allora il primo file viene chiuso ed il secondo prende il controllo.

Se un programma sotto il controllo di EXEC viene interrotto con CTRL-C, generalmente EXEC non continua.

Se un programma sotto il controllo di EXEC contiene una INPUT, l'ingresso viene prelevato dal successivo campo del file EXEC. Questo può causare dei problemi se tale successivo campo contiene un comando DOS in modo immediato (non una linea di programma); il comando verrà eseguito invece di essere accettato come linea in ingresso.

Se un programma esegue un comando CLOSE, o se il file EXEC contiene un comando CLOSE in modo immediato, il file EXEC non viene chiuso.

Uso di EXEC per convertire un programma da un BASIC all'altro

Il comando EXEC può essere usato per convertire un programma dall'Applesoft all'Integer BASIC e viceversa.

Per prima cosa memorizzate il vostro programma come testo di un file. Per fare questo aggiungete al programma le istruzioni di apertura e di scrittura del file; il listato diverrà così il testo del file. Ecco le istruzioni necessarie:

```
 1 POKE 33,33: REM DISABLE "LIST" FORMATTING
 2 D$ = "": PRINT D$"OPEN FILE": PRINT D$;"WRITE FILE"
 3 LIST 10,32767
 4 PRINT D$;"CLOSE": END
 5 POKE 33,40: REM RESET "LIST" FORMATTING
10 REM YOUR PROGRAM BEGINS HERE
```

Quando eseguirete questo programma solo le linee 1, 2 e 3 saranno in effetti eseguite. Esse memorizzeranno nel file il listato del programma. Successivamente cambiate il BASIC (con FP o INT per esempio). Dopodichè battete:

```
EXEC FILE
```

Le linee del vecchio programma vengono allora caricate dal disco nella memoria; se vi sono linee che non sono conformi con il nuovo BASIC dovete però modificarle.

MAXFILES

Il comando MAXFILES permette di stabilire il numero massimo di file che possono essere aperti contemporaneamente.

Ogni file richiede 595 byte di memoria come buffer. Due blocchi di 256 byte sono usati come buffer veri e propri uno per la lettura ed uno per la scrittura del file. I rimanenti 83 byte sono impiegati per tutte le informazioni necessarie alla gestione del file come per esempio gli indirizzi delle tracce e dei settori.

Quando un file viene aperto, oppure vengono eseguiti comandi come CATALOG e LOCK, il DOS legge 256 byte dal disco e li trasferisce nel buffer. Se questa informazione viene cambiata e deve quindi essere riscritta sul file, i 256 byte sono ricopiati sul disco dal buffer.

Il seguente esempio stabilisce che il numero massimo di file è otto:

```
MAXFILES 8
```

Il numero dei file può essere compreso tra 1 e 16. Quando caricate il DOS, vengono automaticamente preparati tre buffer per tre file contemporanei. Se volete lavorare con un numero maggiore di file contemporanei, dovete usare il comando MAXFILES. Se volete risparmiare memoria e aprire meno di tre buffer, dovete egualmente usare il comando MAXFILES.

Un buffer viene sempre impegnato per eseguire i seguenti comandi DOS:

APPEND	FP	POSITION
BLOAD	INIT	READ
BRUN	INT	RENAME
BSAVE	LOAD	RUN
CATALOG	LOCK	SAVE
CHAIN	MAXFILES	UNLOCK
CLOSE	MON	VERIFY
DELETE	NOMON	WRITE
EXEC	OPEN	

Facciamo allora un esempio. Se avete già aperto tutti i file disponibili e date un comando come CATALOG, che richiede un suo buffer, il DOS vi risponde con l'errore NO BUFFER AVAILABLE. Attenzione però che non sono necessari buffer di memoria se i comandi lavorano fuori del contesto DOS (per esempio un LOAD da cassette).

Quando cambiate il numero di file con MAXFILES, i programmi in Integer BASIC vengono cancellati e le stringhe in Applesoft diventano confuse. Per questo motivo eseguite il comando MAXFILES all'inizio prima di eseguire i programmi.

Il comando MAXFILES può essere eseguito in un programma Applesoft se inserito in una PRINT e preceduto da CTRL-D. Se però MAXFILES non è la prima istruzione del programma, può causare dei malfunzionamenti nelle successive istruzioni GOTO, GOSUB e altre. Per evitare di avere dei problemi con le stringhe usate il comando MAXFILES in questo modo:

```
1 REM DARE PRIMA IL COMANDO MAXFILES

2 PRINT CHR$(4);"MAXFILES 9"

3 REM IL PROGRAMMA INIZIA DA QUI

4 D$=CHR$(4): REM CARATTERE CTRL-D
```

In Integer BASIC potete usare MAXFILES solo in un file EXEC. Per esempio un file EXEC contiene queste tre istruzioni in modo pseudo-immediato:

```
MAXFILES 8
LOAD PROGRAM
RUN 10
```

Questo file può essere richiamato da un programma in questo modo:

```
5 PRINT D$;"EXEC MXF": REM PONE MAXFILES,
  CONTINUA ALLA LINEA 10
10 REM IL PROGRAMMA INIZIA QUI
```

USO DEGLI AUSILI DI DEBUGGING DEL DOS

Dal momento che i programmi che gestiscono i file su disco sono abbastanza complessi, il DOS prevede tre comandi speciali per facilitare il loro debugging: MON, NOMON e TRACE.

MON

MON significa "monitor" e permette di controllare le informazioni che sono scambiate con il disco. Il formato di MON prevede l'uso dei tre parametri I, O e C; ecco un esempio:

```
MON C,I,O
```

Il parametro C fa visualizzare i comandi al disco. I fa visualizzare i dati provenienti dal disco. O fa visualizzare i dati inviati al disco.

I tre parametri possono essere dati in qualunque ordine e non necessariamente tutti e tre; almeno uno deve essere indicato, perchè diversamente il comando sarebbe trascurato.

MON rimane valido sin tanto che venga dato un comando NOMON o INT o FP, oppure il DOS stesso venga ricaricato.

NOMON

NOMON cancella gli effetti del comando MON. Se nella lista di NOMON compaiono alcuni dei parametri di MON, allora solo queste funzioni vengono cancellate. Per esempio se dopo MON I, O, C diamo NOMON O, solo la funzione uscita viene cancellata e continuano ad essere visualizzati i comandi al disco e i dati in ingresso.

Il comando TRACE

Il comando TRACE (vedere il capitolo 4) è usato per il debugging dei programmi. Ma poichè il TRACE fa visualizzare i numeri di linea senza ritorno del carrello, i comandi DOS risulterebbero concatenati ai numeri di linea e quindi trascurati. Per rimediare a ciò basta dare un ritorno carrello prima di ogni comando DOS. Per fare questo cambiate la linea:

```
100 D$ =  CHR$ (4): REM  CTRL-D
```

in:

```
100 D$ =  CHR$ (13) +  CHR$(4): REM  RETURN + CTRL
```

In questo modo ogni volta che viene eseguita una PRINT D$, il carattere CTRL-D risulta preceduto da un ritorno del carrello per cui si ha la sua separazione dal numero di linea di TRACE.

Esiste però un altro problema per il quale non sappiamo proprio quali rimedi consigliarvi. Se mentre state facendo il TRACE è presente un comando WRITE, allora tutti i numeri di linea andranno a finire sul file!

FILE SU DISCO IN LINGUAGGIO MACCHINA (IMMAGINI BINARIE)

Con il Disco II potete depositare sui dischetti anche file in linguaggio macchina e immagini binarie (grafici). Questi file sono indicati con la lettera B nella lista di un catalogo.

Sia le immagini a bassa risoluzione che quelle ad alta risoluzione possono essere

memorizzate su un dischetto per poter essere visualizzate in un secondo tempo.

I programmi in linguaggio macchina possono essere caricati ed eseguiti direttamente, oppure possono essere chiamati da un programma BASIC mediante l'istruzione CALL o la funzione USR.

Il DOS prevede tre speciali comandi per gestire i file binari. Essi sono BSAVE, BLOAD e BRUN e sono equivalenti agli analoghi comandi non binari (BSAVE = SAVE, BLOAD = LOAD e BRUN = RUN).

BSAVE

BSAVE salva una immagine binaria sul disco. Ecco un esempio:

```
BSAVE FILENAME,A378,L21,S6,D2,V6
```

Notate che questo comando richiede due parametri *obbligatori* A e L, mentre gli altri sono opzionali.

A è il parametro *indirizzo* e rappresenta l'indirizzo iniziale dell'immagine binaria che viene salvata sul disco. Questo indirizzo di memoria può essere una costante sia decimale che esadecimale. I valori esadecimali devono essere preceduti dal segno di dollaro ($). Quelli decimali devono essere compresi tra O e 65535. Non sono ammessi valori negativi.

Il parametro L indica la *lunghezza* dell'immagine binaria che viene salvata su disco ed è espresso in numero di byte. Anche questo parametro può essere specificato sia in decimale che in esadecimale; come per il parametro A se si usa la forma esadecimale bisogna precederla con il segno di dollaro ($). Il valore decimale può essere compreso tra 1 e 32767; se viene superato si commette l'errore SYNTAX ERROR. 32767 byte costituiscono il più lungo campo che possa essere memorizzato dal DOS. Se si devono memorizzare più di 32767 byte è necessario usare due comandi BSAVE.

Se le indicazioni di memoria superano le disponibilità fisiche del vostro calcolatore, allora non viene segnalato alcun errore. (Per esempio se indicate A49151, o A$BFFF, in un sistema che ha solo 48K di memoria).

BLOAD

BLOAD carica un file binario in memoria. Ecco un esempio di comando BLOAD:

```
BLOAD FILENAME,A378,S6,D2,V6
```

Il parametro indirizzo è opzionale; se manca, l'immagine viene caricata a partire dallo stesso indirizzo specificato quando l'immagine fu salvata con BSAVE.

Attenzione però che i programmi in linguaggio macchina possono non funzionare correttamente se vengono caricati a partire da indirizzi di memoria inesatti.

Ricordatevi inoltre che BLOAD, a differenza di LOAD, non cancella i programmi e i

dati precedenti, ma solo quella zona della memoria in cui viene caricato il nuovo programma.

Se nelle indicazioni di BLOAD specificate posizioni della memoria a sola lettura ROM non vi verrà segnalato alcun errore, ma ovviamente non scriverete niente nella ROM stessa!

BRUN

Il comando BRUN è equivalente a BLOAD salvo che, dopo aver caricato il file, BRUN esegue una istruzione di salto in linguaggio macchina JMP all'indirizzo di partenza del file. Se non viene specificato alcun indirizzo di partenza, il salto viene fatto allo stesso indirizzo iniziale che era stato indicato per salvare il file. Ecco un esempio di comando BRUN:

```
BRUN FILENAME,A378,S6,D2,V6
```

Non usate mai il comando BRUN con le immagini grafiche, perchè potreste ottenere dei risultati assolutamente errati.

CAPITOLO 6

GRAFICI E SUONI

Il calcolatore Apple II ha la possibilità di tracciare sullo schermo dei grafici a colori e di generare dei suoni.

Queste due caratteristiche rendono l'uso del calcolatore Apple II molto versatile ed interessante. Il presente capitolo è dedicato principalmente alle persone che si avvicinano alla programmazione per la prima volta, ma hanno già una conoscenza degli elementi fondamentali del BASIC e del linguaggio assembler.

La gestione dei grafici non è difficile, specialmente nei linguaggi ad alto livello. Il Monitor dell'Apple II, con le sue subroutine, facilita molto la scrittura dei programmi grafici in assembler e la gestione dei suoni. Riteniamo che dopo la lettura di questo capitolo sarete in grado di fare degli ottimi programmi.

GRAFICI A BASSA RISOLUZIONE

Il calcolatore Apple II ha due aree di memoria riservate per i grafici a bassa risoluzione che vengono chiamate *pagine*. Ambedue queste pagine possono essere visualizzate sull'intero schermo (40 colonne per 48 righe). Ogni pagina può essere rappresentata come in Figura 6.1.

Ogni punto definito da una riga, e da una colonna, appare sullo schermo come un piccolo rettangolino. Di questi rettangolini ve ne sono 1920 per ogni pagina (40 x 48) ed ognuno può essere colorato con uno dei 16 colori fondamentali (vedi Tabella 6.1). Per lavorare con i grafici a bassa risoluzione è sufficiente conoscere il BASIC e avere alcune nozioni di geometria.

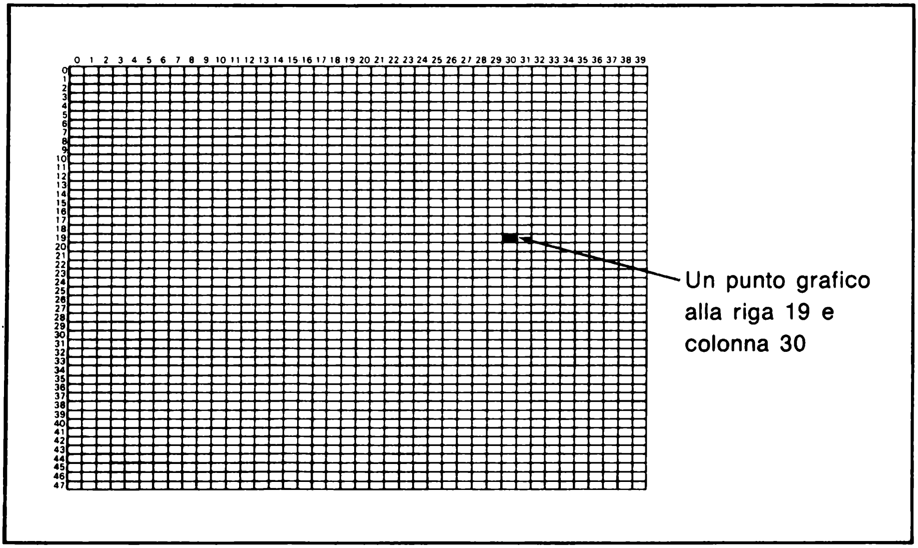


Figura 6.1 — Schermo per grafici a bassa risoluzione.

IMPOSTAZIONE DELLA PAGINA GRAFICA

Per operare nel modo grafico, in BASIC, dovete dare l'istruzione:

```
GR
```

Quando questa istruzione viene eseguita, lo schermo diviene tutto nero eccetto che quattro linee in basso che sono riservate per il testo. Questa parte dello schermo prende il nome di *finestra per il testo.* È ovvio che in tal caso la zona rimasta disponibile per i grafici si è ridotta a 40 righe anzichè 48.

L'istruzione GR può essere usata più di una volta in uno stesso programma grafico a bassa risoluzione per cancellare lo schermo.

Tabella 6.1 — Colori dei grafici a bassa risoluzione.

Colore	Numero	Colore	Numero
Nero	0	Bruno	8
Magenta	1	Arancione	9
Blu scuro	2	Grigio # 2	10
Porpora	3	Rosa	11
Verde scuro	4	Verde chiaro	12
Grigio # 1	5	Giallo	13
Blu medio	6	Acqua	14
Blu chiaro	7	Bianco	15

Grafici su tutto lo schermo

Dopo l'esecuzione dell'istruzione GR, potete eliminare la finestra per il testo, ed utilizzare anche questa parte dello schermo per i grafici, mediante il comando:

```
POKE -16302,0
```

La finestra allora scompare lasciando il posto ai grafici.

Ripristino della finestra per il testo

Se il calcolatore sta lavorando con i grafici su tutto lo schermo, potete ripristinare la finestra per lo schermo in due modi. Potete dare nuovamente il comando GR ottenendo così sia di cancellare lo schermo, sia di far riapparire la finestra per il testo contemporaneamente. Se volete invece fare apparire la finestra senza alterare le prime 40 righe del grafico, date il seguente comando:

```
POKE -16301,0
```

Una volta eseguita, questa istruzione riapre la finestra per il testo sulla parte bassa dello schermo.

Ritorno allo schermo tutto-testo

Per ritornare ad operare con lo schermo tutto-testo, date il comando:

```
TEXT
```

Questo comando è in pratica l'opposto di GR, ma con la differenza che TEXT non cancella lo schermo come invece fa GR.

Ricordiamo che i testi, come i grafici, usano la stessa area di memoria. Dopo l'esecuzione di TEXT potete vedere lo schermo pieno di strani caratteri a causa del fatto che il calcolatore tenta di interpretare come testo dei dati grafici che erano già in memoria. Per cancellare lo schermo in queste condizioni usate la sequenza ESC-@, oppure l'istruzione CALL-936, oppure in Applesoft il comando HOME.

ISTRUZIONI PER PROGRAMMARE I GRAFICI

Sia l'Integer BASIC che l'Applesoft prevedono l'uso di comandi, per grafici a bassa risoluzione, per visualizzare singoli punti colorati sullo schermo, per cambiare i colori e per tracciare linee verticali e orizzontali di varia lunghezza. Questi comandi funzionano però solo per la pagina 1 dei grafici a bassa risoluzione (questa pagina viene anche chiamata primaria). Con la pagina 2 non potete purtroppo usare queste comodissime routine e dovete fare uso dei comandi elementari come PEEK, POKE e CALL.

L'istruzione COLOR

L'istruzione COLOR stabilisce quale deve essere il colore con cui tracciare successivamente dei punti. Ecco un esempio:

```
COLOR=13
```

Il suo parametro è uno dei numeri compresi tra 0 e 15 della Tabella 6.1; nell'esempio viene scelto il colore giallo (13). Se il parametro non viene indicato, il calcolatore impone il colore nero equivalente a COLOR = O. Sebbene possiate porre come parametro un qualunque valore sino a 255, senza generare un errore di sintassi, l'istruzione COLOR accetta solo la *parte inferiore* del numero, compresa tra 0 e 15. In altre parole se ponete COLOR = 222 (cioè DE in esadecimale), esso viene interpretato come COLOR = 15 (cioè OE in esadecimale).

L'istruzione PLOT

Con questa istruzione fate apparire un punto (cioè un rettangolino) sullo schermo alle coordinate indicate. Per esempio l'istruzione:

```
PLOT 23,18
```

abilita un punto sulla 24-esima riga e 19-esima colonna con il colore stabilito dalla precedente istruzione COLOR. I numeri di riga vanno da 0 a 47; quelli di colonna da 0 a 39. Se superate questi limiti vi viene segnalato un errore e il programma si ferma. Come per tutte le istruzioni grafiche in bassa risoluzione (eccetto GR) potete usare espressioni e variabili per indicare i parametri:

```
PLOT Y/2+12,X-4
```

Un esempio di tracciamento

Il seguente esempio, in Integer BASIC, usa tutte le istruzioni grafiche a bassa risoluzione finora descritte. Esso traccia una diagonale dall'angolo sinistro alto verso destra in basso:

```
10  REM TRACCIA UNA LINEA DIAGONALE
11  REM SULLO SCHERMO IN BASSA RISOLUZ.
20  GR
30  COLOR=  RND (16)
40  FOR Y = 0 TO 39
50  PLOT Y,Y
60  NEXT Y
70  GOTO 30
```

Il risultato è riportato in Figura 6.2 e il suo colore varierà in modo ràndom.

Per convertire questo programma in Applesoft, cambiate la linea 30 in questo modo:

```
30 COLOR= RND (16) * 16
```

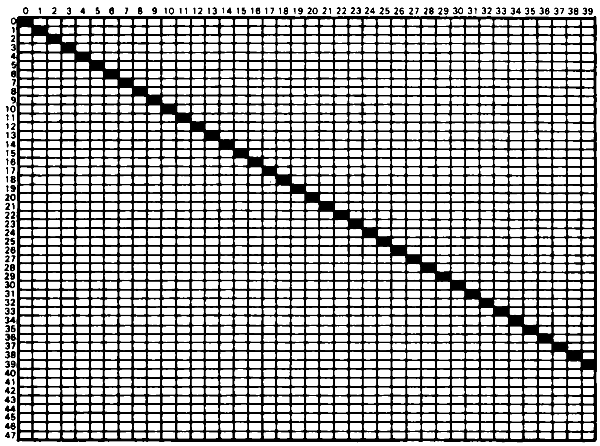


Figura 6.2 — Esempio di linea a bassa risoluzione.

Tracciamento di linee orizzontali

Il comando HLIN permette di tracciare linee orizzontali, da sinistra verso destra con lunghezza variabile, sulla pagina grafica a bassa risoluzione. In questo esempio:

```
HLIN 0,39 AT 0
```

viene tracciata una linea sulla riga più in alto dello schermo dalla prima all'ultima colonna. I parametri di HLIN sono tre: colonna iniziale e finale della linea da tracciare e (AT) riga su cui tracciarla. Nell'esempio la linea è tracciata dalla colonna 0 alla colonna 39 sulla riga 0.

I due numeri, che indicano gli estremi della linea, non possono essere negativi e devono essere compresi tra 0 e 255. Il secondo non può essere inferiore al primo. Il parametro di riga non può essere negativo ne superiore a 48. Se uno di questi parametri è posto oltre i suoi limiti, vi viene segnalato un errore e il programma si ferma. In Integer BASIC se ponete i numeri estremi di colonna della linea superiori a 39, potete ottenere dei risultati assolutamente imprevedibili. In Applesoft ottenete invece una segnalazione di errore. Se per esempio eseguite:

```
10 GR

20 COLOR=12

30 HLIN 45,100 AT 0
```

in Integer BASIC vi appaiono due linee verdi. La prima non è sulla riga 0, come dovrebbe essere, ma continua alcune righe più in basso. Attenzione quindi a porre sempre i parametri giusti!

Tracciamento di linee verticali

Il comando VLIN traccia una linea verticale, con il colore già selezionato, da una riga ad un'altra su una certa colonna.

Per esempio:

```
VLIN 12,30 AT 33
```

traccia una linea verticale dalla riga 12 alla riga 30 sulla colonna 33. I parametri di VLIN devono soddisfare alle stesse condizioni di quelli di HLIN. I primi due devono essere valori interi compresi tra 0 e 255; il secondo non può essere inferiore al primo. Il terzo parametro, corrispondente al numero di colonna, deve essere tra 0 e 39. Se uno di essi è posto fuori dei valori consentiti, vi verrà segnalato un messaggio di errore.

Uso di HLIN e VLIN in un programma

Il programma seguente è un esempio di come usare le istruzioni HLIN e VLIN. Per dare un maggiore effetto, lasciamo che la scelta dei colori, e la posizione delle linee, siano stabilite dalla funzione random per la generazione dei numeri a caso. Per fermare il programma battete CTRL-C.

```
10  REM DIMOSTRAZIONE IN BASSA RISOLUZ.
11  REM DI HLIN E VLIN
20  GR : REM PAGINA GRAFICA 1
30  POKE  - 16302,0: REM  PAGINA INTERA
40  CALL  - 1998: REM PULISCE TUTTE LE 48 RIGHE
50  REM INIZIO PROGRAMMA
60  COLOR=  RND (16): REM COLORE SCELTO A CSO
70  HLIN 0, RND (40) AT  RND (48)
80  COLOR=  RND (16)
90  VLIN 0, RND (48) AT  RND (40)
100  GOTO 60
```

L'istruzione SCRN

L'uso di questa istruzione è meno facilmente intuibile delle precedenti, ma vi sarà molto utile quando scriverete programmi grafici a bassa risoluzione piuttosto complessi.

L'istruzione SCRN informa l'operatore, o il programma, quale colore è in quel momento presente su un certo punto dello schermo. Per esempio:

```
X=SCRN(12,24)
```

assegna il valore numerico del colore, del punto di coordinate 12 e 24, alla variabile X. In altre parole viene riconosciuto il codice di colore, secondo la Tabella 6.1, e passato alla variabile X. Per esempio con le istruzioni:

```
GR
COLOR=14
PLOT 12,12
PRINT SCRN (12,12)
```

il calcolatore risponde: 14.

GRAFICI AD ALTA RISOLUZIONE

Una delle caratteristiche più importanti del calcolatore APPLE II è la sua possibilità di tracciare grafici ad alta risoluzione.

Anche in questo caso sono disponibili due pagine di memoria, ma con una ben maggiore risoluzione: 280 posizioni orizzontali per 192 posizioni verticali. Cioè 7 volte più risoluzione sull'asse orizzontale e 4 volte di più su quello verticale.

I colori disponibili sono solo otto, ma in compenso si possono tracciare i disegni con ben maggiore chiarezza.

Funzioni grafiche ad altra risoluzione sono disponibili solo in Applesoft. L'Integer BASIC non ha delle vere e proprie funzioni grafiche di questo tipo, ma qualunque sia il linguaggio che usate, l'Apple II ha sempre la possibilità di gestire grafici ad alta risoluzione. Infatti il calcolatore Apple II usa parte della sua memoria per memorizzare in alta risoluzione punti, linee, curve e speciali programmi per interpretare questi dati e visualizzarli sullo schermo. Alcune istruzioni Applesoft usano direttamente questi programmi e di ciò ne parleremo subito. Successivamente vedremo come usare i grafici ad alta risoluzione in Integer BASIC. Infine vi insegneremo qualche trucco, in alta risoluzione, che potrete usare *solo* in Integer BASIC.

QUALE PAGINA USARE?

L'uso dei grafici ad alta risoluzione dipende all'ammontare di memoria che il vostro calcolatore possiede.

Se avete l'Applesoft su cartolina (cioè nelle ROM oppure sulla cartolina "Apple II Language System") potete usare la pagina 1 ad alta risoluzione solo *se* il vostro calcolatore ha più di 16K di memoria; potete usare inoltre la pagina 2 *se* avete almeno 24K di memoria. Se poi contemporaneamente ai grafici ad alta risoluzione volete usare anche il DOS allora dovete prevedere altri 12K di memoria.

Se invece il vostro Applesoft proviene da cassetta o da disco, non potete usare la pagina 1 ad alta risoluzione, perchè essa contiene parte oell'interprete Applesoft. Se tentate di usare la pagina 1 rischiate di danneggiare, o perdere, l'interprete Applesoft

e il programma in esecuzione in memoria. *Potete* usare però la pagina 2 se avete almeno 24K di memoria; se volete lavorare contemporaneamente anche con il DOS dovete avere allora almeno 36K di memoria.

Sconfinamento nell'area grafica ad alta risoluzione

L'Applesoft non protegge automaticamente la zona di memoria che contiene i grafici ad alta risoluzione. Se un programma viene troppo ampliato può succedere che sconfini nell'area riservata ai grafici rovinandoli. Per ovviare a questo si possono usare i due puntatori HIMEM: e LOMEM: in modo da proteggere l'area dei grafici.

Questi puntatori funzionano così da confine per i programmi BASIC che non li possono superare.

L'uso di HIMEM: e LOMEM: differisce tra l'Applesoft e l'Integer BASIC. Per comprendere più facilmente quanto diremo ora, consultate prima le voci HIMEM: e LOMEM: del capitolo 8 e la figura G.1 dell'Appendice G.

Se volete usare la pagina 1 ad alta risoluzione (ricordatevi che però dovete avere l'Applesoft su cartolina) e se avete solo 16K di memoria, ponete allora HIMEM: eguale a 8191 e lasciate libero LOMEM:. Di conseguenza il vostro programma non potrà superare la posizione di memoria 8191; questa limitazione può sembrare molto restrittiva, ma è assolutamente necessaria perchè avete bisogno di 8K per gestire i grafici ad alta risoluzione. Se poi avete più di 16K di memoria allora è preferibile che lasciate libero HIMEM: e poniate LOMEM: eguale a 16384. In tal caso il vostro programma Applesoft viene posto sopra la pagina 1 ad alta risoluzione.

ATTENZIONE: non usate quest'ultima soluzione se volete lavorare anche con il DOS e non avete almeno 32K di memoria. È preferibile, in questo caso, che scegliate la prima soluzione ponendo il programma sotto la pagina 1.

Vediamo ora che cosa fare per lavorare con la pagina 2 ad alta risoluzione. In questo caso (se avete almeno 24K di memoria) ponete LOMEM: eguale a 24576, oppure ponete HIMEM: eguale a 16383 e lasciate LOMEM: invariato.

Se ponete il valore di LOMEM: voi portate il vostro programma Applesoft *sopra* la pagina 2 ad alta risoluzione. Per usare il DOS contemporaneamente, avete bisogno però di almeno 48K di memoria. Se imponete invece il valore di HIMEM: il programma risiederà *sotto* la pagina 2. In quest'ultimo caso se usate un interprete Applesoft proveniente da cassetta o disco, il vostro programma dovrà essere compresso tra l'interprete sottostante e la pagina 2 soprastante.

Se infine pensate di usare ambedue le pagine (il chè vi è possibile solo se avete l'Applesoft su cartolina) ponete allora LOMEM: eguale a 24576 oppure HIMEM: eguale a 8191. Se ponete il vostro programma sopra alle pagine ad alta risoluzione, aggiustando il valore di LOMEM: e lasciando libero quello di HIMEM:, allora avrete bisogno di almeno 48K di memoria per potere usare il DOS ed avere ancora sufficiente memoria per i vostri programmi.

VISUALIZZAZIONE DEI GRAFICI

Sebbene le pagine grafiche siano due, la prima non è utilizzabile se avete l'interprete Applesoft nella versione a cassetta o a disco. Infatti l'interprete stesso occupa una parte della prima pagina. Se avete invece un calcolatore Apple II Plus o Apple II standard, dotato dell'Applesoft su cartolina o della cartolina Language System, potete usare la pagina 1 senza problemi.

Spesso per compatibilità tra i vari calcolatori Apple II è preferibile usare sempre la pagina 2 così da rendere i programmi indipendenti dal tipo di Applesoft.

L'istruzione:

```
HGR
```

dell'Applesoft cancella e visualizza la pagina 1 ad alta risoluzione lasciando sulla parte bassa dello schermo una finestra per il testo di quattro linee. In questo modo potete avere contemporaneamente sullo schermo un grafico e un testo. Attenzione però che il grafico arriva sino alla riga 160 perchè lascia le altre, sino alla riga 192, disponibili per la finestra. Per avere il grafico su tutto lo schermo dovete dare il comando POKE −16302,0 dopo HGR.

Per visualizzare la pagina 2 dovete usare l'istruzione:

```
HGR2
```

Questa istruzione cancella e visualizza la pagina 2, senza però lasciare alcuna finestra sullo schermo coma fa invece HGR. Se volete aprire la finestra per il testo dovete dare il comando POKE −16301,0 dopo HGR2.

È importante precisare però che la scrittura nella finestra della pagina 2 è più complicata di quella della pagina 1. Infatti per scrivere nella finestra della pagina 1 si possono usare le normali istruzioni PRINT, mentre per scrivere in quella della pagina 2 si può solo usare il comando POKE che è molto più laborioso. Inoltre la pagina di testo secondaria non è protetta dagli sconfinamenti del BASIC, come lo è invece la pagina di testo 1; per proteggerla dovete quindi imporre LOMEM: eguale a 3071 o ad un valore superiore.

POSSIBILI ALTERNATIVE AD HGR E HGR2

Il maggior svantaggio nell'uso di HGR e HGR2 è quello che esse causano la cancellazione della pagina visualizzata. Inoltre queste due istruzioni non sono disponibili in Integer BASIC.

Invece di HGR e HGR2 potete usare le istruzioni PEEK, POKE e CALL e alla fine ottenere dei risultati altrettanto validi e pratici.

Un altro modo per visualizzare i grafici

Vi illustriamo come sia possibile lavorare con i grafici ad alta risoluzione senza do-

vere necessariamente cancellare le pagine grafiche in memoria. Per comprendere questo vi dobbiamo parlare degli *interruttori a software* posti nelle posizioni di memoria da −16304 a −16297 (da $C050 a $C057). Essi non sono altro che delle particolari posizioni di memoria che contengono delle informazioni come le potrebbero contenere un insieme di veri interruttori. Nell'Appendice E vi diamo una descrizione più dettagliata degli interruttori a software. Gli indirizzi di questi interruttori sono dati in forma intera negativa per rendere le istruzioni accettabile sia dall'Applesoft che dall'Integer BASIC (usiamo per esempio −16304 invece di 49231).

Per visualizzare la pagina 1 ad alta risoluzione senza cancellarne il contenuto, eseguite le seguenti istruzioni:

POKE −16304,0	Scelta del modo grafico
POKE −16297,0	Scelta dei grafici ad alta risoluzione
POKE −16300,0	Seleziona la pagina ad alta risoluzione 1 (Questa istruzione è necessaria se precedentemente lavoravate con la pagina 2)

Provate per esercizio a dare queste istruzioni in modo immediato.

Se volete invece visualizzare la pagina grafica 2 ad alta risoluzione, senza cancellarne il contenuto, battete queste istruzioni:

POKE −16304,0	Scelta del modo grafico (se non già scelto in precedenza)
POKE −16297,0	Scelta dei grafici al alta risoluzione (se non stabilito già in precedenza)
POKE −16299,0	Seleziona la pagina ad alta risoluzione 2

Ritorno al BASIC dopo i grafici ad alta risoluzione

In Integer BASIC le istruzioni TEXT e GR possono non essere sufficienti per ripristinare lo schermo per l'uso normale. Se stavate lavorando con la pagina grafica 2 dovete prima esplicitamente selezionare la pagina 1 con l'istruzione POKE −16300,0; diversamente potreste vedere apparire la pagina 2 a bassa risoluzione. Per esempio se battete qualcosa alla tastiera potreste trovarvi molto disorientati, perchè ciò che battete va sulla pagina 1, mentre invece sullo schermo vi appare la pagina 2; in altre parole potrebbe sembrarvi che la tastiera sia guasta.

L'istruzione GR non fa passare dai grafici ad alta risoluzione a quelli a bassa risoluzione in Integer BASIC. Essa seleziona unicamente il modo grafico con una finestra per il testo di quattro linee (a differenza di tutti i modi che trattano testi). Dovete esplicitamente selezionare i grafici a bassa risoluzione con l'istruzione POKE −16298,0.

Cancellazione delle pagine ad alta risoluzione

Se usate le istruzioni HGR e HGR2 in Applesoft ottenete la cancellazione automati-

ca della pagina selezionata. In Integer BASIC non esistono invece questi comandi per cui è necessario impiegare una routine, che usa una funzione del Monitor, per cancellare lo schermo dai grafici ad alta risoluzione. Ecco la routine:

```
18990 REM  ****************************
18991 REM  *   CLEAR HI-RES SCREEN    *
18992 REM  * (USES MONITOR'S "MOVE"   *
18993 REM  *  SUBROUTINE AT $FE2C     *
18994 REM  * TO MOVE DATA QUICKLY     *
18995 REM  * THROUGH THE HI-RES AREA)*
18996 REM  * -------------------------*
18997 REM  * SET PAGE=1 OR 2; THE     *
18998 REM  * ROUTINE DOES THE REST.   *
18999 REM  ****************************
19000 START=32
19010 IF PAGE=2 THEN START=64
19020 POKE 60,0
19030 POKE 61,START
19040 POKE 62,254
19050 POKE 63,START+33
19060 POKE 66,1
19070 POKE 67,START
19080 POKE -16304,0
19090 POKE -16297,0
19100 IF PAGE=1 THEN POKE -16300,0
19110 IF PAGE=2 THEN POKE -16299,0
19120 POKE START*256,0
19130 CALL -468
19140 RETURN
```

Essenzialmente questa routine riempie di zeri la pagina grafica ad alta risoluzione selezionata. Se la variabile PAGE è posta eguale ad 1, viene cancellata la pagina 1; se PAGE è uguale a 2 viene cancellata la pagina 2.

Questa routine lavora solo in Integer BASIC; in Applesoft si possono usare invece le istruzioni HGR e HGR2.

COLORI DEI GRAFICI AD ALTA RISOLUZIONE

I grafici ad alta risoluzione possono scegliere tra otto codici di colore anche se i colori veramente diversi sono solo quattro più il bianco e il nero. In Tabella 6.2 sono riportati questi colori e i loro corrispondenti codici.

L'istruzione HCOLOR

L'istruzione HCOLOR, disponibile solo in Applesoft, seleziona uno degli otto colori disponibili per l'alta risoluzione. A differenza dell'istruzione COLOR, valida per i grafici a bassa risoluzione, l'istruzione HCOLOR non accetta numeri di colore superiori a 7 (COLOR ammette invece numeri di colore superiori a quelli disponibili). Se indicate

un numero di colore non consentito, il programma si ferma e vi appare il messaggio di errore? ILLEGAL QUANTITY ERROR. HCOLOR non cambia il colore di grafici che siano già presenti sullo schermo, ne ha alcun effetto sui grafici a bassa risoluzione.

Tabella 6.2 – Colori dei grafici ad alta risoluzione

Colore	Numero	Colore	Numero
Nero	0	Nero	4
Verde	1	Arancione	5
Violetto	2	Blu	6
Bianco	3	Bianco	7

Come impostare un colore di fondo in alta risoluzione

La routine di cancellazione che abbiamo descritto può essere usata, con piccole modifiche, per riempire lo schermo con un colore uniforme scelto tra quelli consentiti per i grafici ad alta risoluzione. Prima di richiamare la routine è necessario stabilire su quale pagina lavorare attribuendo un valore a PAGE e scegliere il colore dando un valore alla variabile HCOLOR. Il programma principale chiamante deve anche porre i valori, per gli interruttori a software, per lavorare in alta risoluzione. Ricordatevi poi che questa routine lavora solo in Integer BASIC:

```
18990 REM  *********************
18991 REM  * SET BACKGRND COLOR*
18992 REM  *-------------------*
18993 REM  * SET PAGE=1 OR 2;  *
18994 REM  * ALSO SET HCOLR TO *
18995 REM  * APPLESOFT EQUIVA- *
18996 REM  * LENT HI-RES COLOR.*
18997 REM  *********************
19000 START=32
19010 IF PAGE=2 THEN START=64
19020 POKE 60,0
19030 POKE .61,START
19040 POKE 62,253
19050 POKE 63,START+33
19060 POKE 66,2
19070 POKE 67,START
19075 Y1=85:X1=42
19080 IF HCOLR>0 AND HCOLR<>4 THEN 19090:
19090 IF HCOLR<>3 AND HCOLR<>7 THEN 1910(
19100 IF HCOLR=1 OR HCOLR=5 THEN 19120
19110 X3=X1:X1=Y1:Y1=X3
19120 IF HCOLR<4 THEN 19140
19130 Y1=Y1+128:X1=X1+128
19140 POKE START*256,X1
19150 POKE START*256+1,Y1
```

```
19160 IF PAGE=1 THEN POKE -16300,0
19170 IF PAGE=2 THEN POKE -16299,0
19180 CALL -468
19190 RETURN
```

TRACCIAMENTO DI PUNTI E LINEE

In Applesoft è possibile tracciare, nei grafici ad alta risoluzione, linee con qualunque inclinazione e punti. L'istruzione HPLOT può essere usata in tre modi:

```
HPLOT 12,12
```

In questo caso viene disegnato un singolo punto all'incrocio della 13-esima colonna con la 13-esima riga. La pagina e il colore sono quelli già scelti in precedenza.

Il secondo uso di HPLOT è:

```
HPLOT 0,0 TO 279,191
```

Viene tracciata una linea diagonale tra il punto 0,0 e quello 279,191. Con questo formato di HPLOT possono essere tracciate linee tra qualunque due punti dello schermo.

Il terzo caso è più complesso:

```
HPLOT 0,0 TO 279,0 TO 279,191 TO 0,191 TO 0,0
```

È possibile, in questo caso, tracciare più linee facendo uso di una sola istruzione HPLOT. Questa versione di HPLOT funziona solo con l'Applesoft residente su cartolina; manca nel caso di Applesoft caricato da cassetta o da disco. Tutte le linee sono disegnate nello stesso colore.

Soluzioni alternative a HPLOT

La subroutine, che riportiamo qui di seguito, permette di programmare grafici ad alta risoluzione in Integer BASIC senza dover cambiare le pagine. Sebbene sia molto interessante, essa è piuttosto lenta perchè tutti i calcoli sono fatti in BASIC.

Prima di richiamarla dovete stabilire le coordinate X e Y del punto che volete tracciare e assicurarvi che la variabile PAGE corrisponda alla pagina ad alta risoluzione che volete usare. Scegliete infine se lavorare con i grafici ad alta risoluzione e se usare tutto lo schermo.

Se usate questa subroutine *senza* prima avere scelto l'alta risoluzione, essa traccerà il grafico nella pagina ad alta risoluzione, ma non userà lo schermo. Successivamente quando il programma avrà finito di disegnare il grafico nella pagina di memoria, sarà possibile passare all'alta risoluzione con dei comandi POKE (come abbiamo già descritto) ed ottenere subito l'immagine sullo schermo. Questo fatto è molto im-

portante perchè vi permette di disegnare la pagina 2 mentre sullo schermo è visibile la pagina 1 ad alta risoluzione e viceversa.

```
20000 REM  ************************
20001 REM  * HI-RES INTEGER PLOT *
20002 REM  * ------------------- *
20003 REM  * SET X=COL, Y=ROW,    *
20004 REM  * PAGE=1 OR 2; USES    *
20005 REM  * VARS Y1,X1,X3        *
20006 REM  ************************
20007 REM
20010 Y1=PAGE*8192: REM   SET BASE ADDRESS
20020 Y1=Y1+(Y/64)*40+(Y MOD 8)*1024
20030 Y1=Y1+(Y MOD 64/8)*128+X/7
20040 X1= PEEK (Y1): REM  READ IN THE HI-RES BYTE
20050 X3=2 ^ (X MOD 7)
20060 REM  'OR' THE BYTE IN X1 WITH
20070 REM  THE BIT VALUE IN X3.
20080 IF X1 MOD (X3 ^ 2)<X3 THEN X1=X1+X3
20090 POKE Y1,X3
21000 X1= PEEK (Y1)
21010 X3=2 ^ (X MOD 7)
21020 IF X1 MOD (X3*2)<X3 THEN X1=X1+X3
21025 X3=2 ^ (X MOD 7)
21026 GOTO 21100
21030 POKE Y1,X3
21040 RETURN
21100 POKE Y1,X3
21130 RETURN
25000 GOSUB 19000
```

Un esempio di grafico ad alta risoluzione in Integer BASIC

L'esempio che vi presentiamo usa due delle subroutine, appena descritte, per tracciare dei punti sullo schermo con alta risoluzione. I controllori esterni per giochi determinano le coordinate dei punti da tracciare; infatti ruotando le loro manopole si possono tracciare delle linee sullo schermo.

```
 10 REM  THIS PROGRAM USES SPECIAL
 20 REM  SUBROUTINES TO CLEAR AND
 30 REM  PLOT IN HI-RES GRAPHICS
 40 REM  USE CTRL-C TO END PROGRAM
 89 REM  SET GRAPHICS MODE
 90 POKE -16304,0
 99 REM  SET HI-RES GRAPHICS
100 POKE -16297,0
109 REM  SELECT FULL-SCREEN GRAPHICS
110 POKE -16302,0
200 PAGE=2
204 REM  CLEAR HI-RES SCREEN MEMORY
```

```
205 GOSUB 19000
209 REM  GET POINT COORDINATES
210 X= PDL (1)
220 Y= PDL (0)
229 REM  PLOT HI-RES POINT
230 GOSUB 20010
239 REM  GET MORE COORDINATES
240 GOTO 210
260 END
```

Per esercizio provate a fare queste variazioni: usate la subroutine che pone un colore di fondo sullo schermo, invece di cancellarlo; fate un test sui controllori per i giochi con le istruzioni POKE in modo da cancellare lo schermo ogni volta che viene toccato un loro pulsante.

USO DI FIGURE IN ALTA RISOLUZIONE

Il calcolatore Apple II vi permette di definire, disegnare ed elaborare figure bi-dimensionali nei grafici ad alta risoluzione.

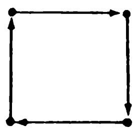
In questo capitolo vediamo come lavorare con queste figure, ma lasciamo alla fantasia del lettore di creare figure e disegni concreti.

Forse avrete già incontrato il problema di ruotare una figura geometrica intorno ad un asse o di ingrandirla e ridurla, in effetti ora vedremo proprio come fare questo genere di elaborazione grafica.

DEFINIZIONE DELLE FIGURE

Tracciare delle figure in alta risoluzione richiede essenzialmente che voi *pianificate* il lavoro di tracciamento delle figure stesse. Non sarà sufficiente che diate dei semplici comandi come disegna il punto A o il punto B, ma dovrete prima preparare una *tavola delle figure* in cui viene codificato tutto il procedimento che poi porterà all'effettiva visualizzazione delle figure.

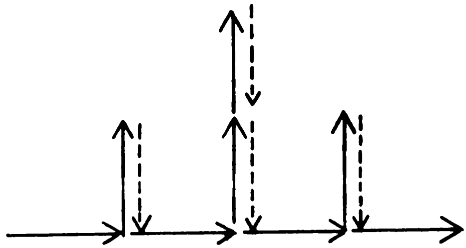
Per costruire la tavola di una figura, iniziate con disegnare la figura su un foglio di carta. Prendiamo per esempio il tracciamento di un quadrato che consiste nel disegnare quattro linee eguali ognuna ad angolo retto rispetto alla precedente:



La tavola conterrà le istruzioni in codice per tracciare i singoli lati del quadrato; tali istruzioni prendono il nome di *vettori grafici.* Ogni vettore rappresenta una operazione

grafica elementare come tracciare una linea orizzontale, una verticale, spostarsi un po' a destra, tracciare un'altra linea, ecc.

Se una figura contiene linee curve o tratti obliqui diviene più complicata, cioè la sua tavola risulterà più lunga. Nella figura seguente si vede, per esempio, quali sono i vettori grafici per tracciare un particolare disegno. Le linee tratteggiate rappresentano le operazioni elementari di solo spostamento e si chiamano *vettori ausiliari:*



Attenzione però! Siccome i vettori grafici possono essere solo orizzontali o verticali ne deriva che i tratti curvi di una figura devono essere approssimati con una successione, tipo gradinata, di tanti vettori. Il disegno finale non può quindi apparire molto preciso per cui talvolta è preferibile usare l'istruzione HPLOT per rappresentare queste curve.

PREPARAZIONE DELLE TAVOLE DELLE FIGURE

Ritorniamo alla figura che avete disegnato su un foglio di carta. È ora necessario scomporla nelle operazioni elementari che saranno poi codificate e inserite nella tavola delle figure. In questo paragrafo vediamo come fare questa conversione, mentre nel prossimo vedremo un programma capace di farla automaticamente.

I codici dei vettori grafici sono compresi tra 0 e 7; tre codici alla volta possono essere raggruppati e inseriti in un byte della tavola delle figure. Nella Tabella 6.3 sono riportati i codici dei vettori grafici.

Preparare una tavola delle figure significa quindi scomporre una figura in tanti vettori grafici elementari, poi codificarli ed inserirli nella tavola la quale sarà poi caricata in memoria. Alcuni comandi speciali Applesoft interpreteranno poi questa tavola e tracceranno sullo schermo la figura originale.

Prendete quindi un punto iniziale della vostra figura e fate una lista di tutti i vettori necessari per ricostruirla con le freccette ↑ ← ↓ →. Non preoccupatevi di seguire la vostra figura in senso orario o antiorario perché non fa alcuna differenza. Segnate quei vettori che rappresentano solo uno spostamento (vettori ausiliari).

Nel caso del quadrato se partiamo dall'angolo di sinistra in basso otteniamo questa lista:

Direzione	Tracciamento
↑	Sì
→	Sì
↓	Sì
←	Sì

Codifichiamo i singoli vettori mediante la Tabella 6.3:

Vettore	Codice
↑	100
→	101
↓	110
←	111

Adesso possiamo raggruppare i codici a tre alla volta in byte separati. Prima di procedere dobbiamo però fare alcune considerazioni. Nella Tabella 6.4 si vede che ogni byte, della tavola delle figure, può essere ripartito in tre sezioni ognuna delle quali può contenere un vettore grafico. Notate però che le sezioni 1 e 2 contengono tre bit ognuna, mentre la terza ha solo due bit.

Tabella 6.3 — Vettori grafici e loro codici binari.

Simbolo	Funzione	Codice binario	Codice decimale
⇡ (tratteggiato)	Muove in su senza tracciare	000	0
⇢ (tratteggiato)	Muove a destra senza tracciare	001	1
⇣ (tratteggiato)	Muove in basso senza tracciare	010	2
⇠ (tratteggiato)	Muove a sinistra senza tracciare	011	3
↑	Muove in su con tracciamento	100	4
→	Muove a destra con tracciamento	101	5
↓	Muove in giù con tracciamento	110	6
←	Muove a sinistra con tracciamento	111	7

Tabella 6.4 — Byte della tavola delle figure.

Bit	Sezione 3		Sezione 2			Sezione 1		
	7	6	5	4	3	2	1	0
M = Bit di movimento P = Bit di tracciamento	M	M	P	M	M	P	M	M

Le prime due sezioni possono così contenere qualunque codice dei vettori, mentre la terza può contenere solo i primi quattro vettori: cioè tutti e quattro i vettori solo di spostamento che hanno il bit di sinistra nullo.

Per questo motivo molte volte troverete che la terza sezione non è usata, anche se potrebbe esserlo. In questi casi la sezione tre viene azzerata e quindi l'Applesoft la ignora e passa avanti ad esaminare il byte successivo.

Attenzione però che porre un codice eguale a zero può significare due cose. Se la sezione tre è posta eguale a zero può significare che deve essere trascurata cioè "nessun tracciamento e nessun spostamento" deve essere eseguito. Ma è anche possibile che essa significhi "spostamento in su senza tracciamento" come indicato nella Tabella 6.3. È ovvio quindi che questa ambiguità può causare notevoli inconvenienti per cui deve essere assolutamente chiarita. La regola che toglie tale ambiguità è la seguente: l'Applesoft interpreta sempre come "non spostamento e non tracciamento" la sezione tre nulla o le sezioni due e tre assieme nulle. Se volete quindi fare eseguire il codice 0 non dovete farlo comparire come terzo elemento di un byte oppure come seconda sezione se anche la terza è nulla.

Se tutte e tre le sezioni sono nulle l'Applesoft interpreta questo come "segno di fine" di una tavola di figure. È ovvio che si debba stabilire uno speciale segno di fine della tavola (cioè il byte tutto nullo) perchè diversamente l'Applesoft continuerebbe ad interpretare, come elementi della figura, i byte posti in memoria dopo la tavola.

Ripetiamo ancora che potete usare il codice 0 "spostamento in su senza tracciamento" purchè nello stesso byte vi sia un codice successivo diverso. Per esempio la sezione 2 può essere posta eguale a 0, mentre la sezione 3 può avere il valore 01 o 10 o 11. Analogamente la sezione 1 può contenere il codice 0, ma per interpretarlo come "spostamento in su senza tracciamento" dovete porre un codice non nullo in una sezione successiva.

A questo punto possiamo finalmente costruire i singoli byte di una tavola e memorizzarli. Nella Tabella 6.5 trovate la tavola corrispondente al quadrato di cui si è parlato precedentemente.

Una volta che i byte sono espressi in forma binaria, potete facilmente convertirli in esadecimale (per comodità consultate le tavole di conversione binario-esadecimale

dell'Appendice J). Nella Tabella 6.5 trovate anche la conversione in esadecimale dei byte del quadrato.

Nel prossimo paragrafo vedremo come sia possibile inserire più figure in una tavola ognuna individuata da un puntatore e tutte elencate in una directory.

Preparazione della directory delle figure di una tavola

La directory di una tavola di figure è costituita da una serie di byte che indicano quante figure ci sono nella tavola e puntano ad ognuna di esse.

Il primo byte della directory contiene il numero totale di figure nella tavola (da 0 a 255, cioè $FF). Il secondo byte non è mai usato ed è posto eguale a zero.

Gli altri byte della directory contengono i puntatori alle figure della tavola. Ogni puntatore è costituito da due byte e contiene lo spostamento (distanza assoluta in byte) tra *l'inizio* della directory e la figura. Il byte di ordine basso del puntatore precede quello di ordine alto. Per esempio se lo spostamento di una figura è di 10 byte il puntatore corrispondente sarà 0A 00 in esadecimale. Nel caso dell'esempio del quadrato vi è una sola figura nella tavola, per cui lo spostamento di tale unica figura dall'inizio della directory è di quattro byte; il puntatore contiene quindi il valore 04 00.

Byte		
0	01	← Numero di figure nella tavola
1	02	
2	04	} ← Spostamento della figura 1 dal byte 0 (il byte di ordine basso è posto prima)
3	00	
4	25	} ← Definizione della figura
5	37	
6	00	← Termine della tavola

Quando impostate la directory di una tavola di figure, è buona norma lasciare sempre dello spazio libero per i puntatori di eventuali nuove figure che in seguito vorrete aggiungere. Se non avete lasciato dello spazio libero nella directory sarete costretti a riscriverla completamente per inserirvi i nuovi puntatori. Se lasciate liberi 20 byte, per esempio, potrete in seguito inserire 10 nuovi puntatori.

Se volete aggiungere una nuova figura ad una tavola, ponetela subito dopo l'ultima; calcolate poi il suo spostamento a partire dall'inizio della directory; inserite quindi tale valore nel nuovo puntatore, che è posto subito dopo l'ultimo puntatore precedente e infine aggiungete 1 al numero totale di figure contenuto del byte 0 della directory.

In Figura 6.3 viene data una rappresentazione di come è organizzata una tavola di figure e la sua directory.

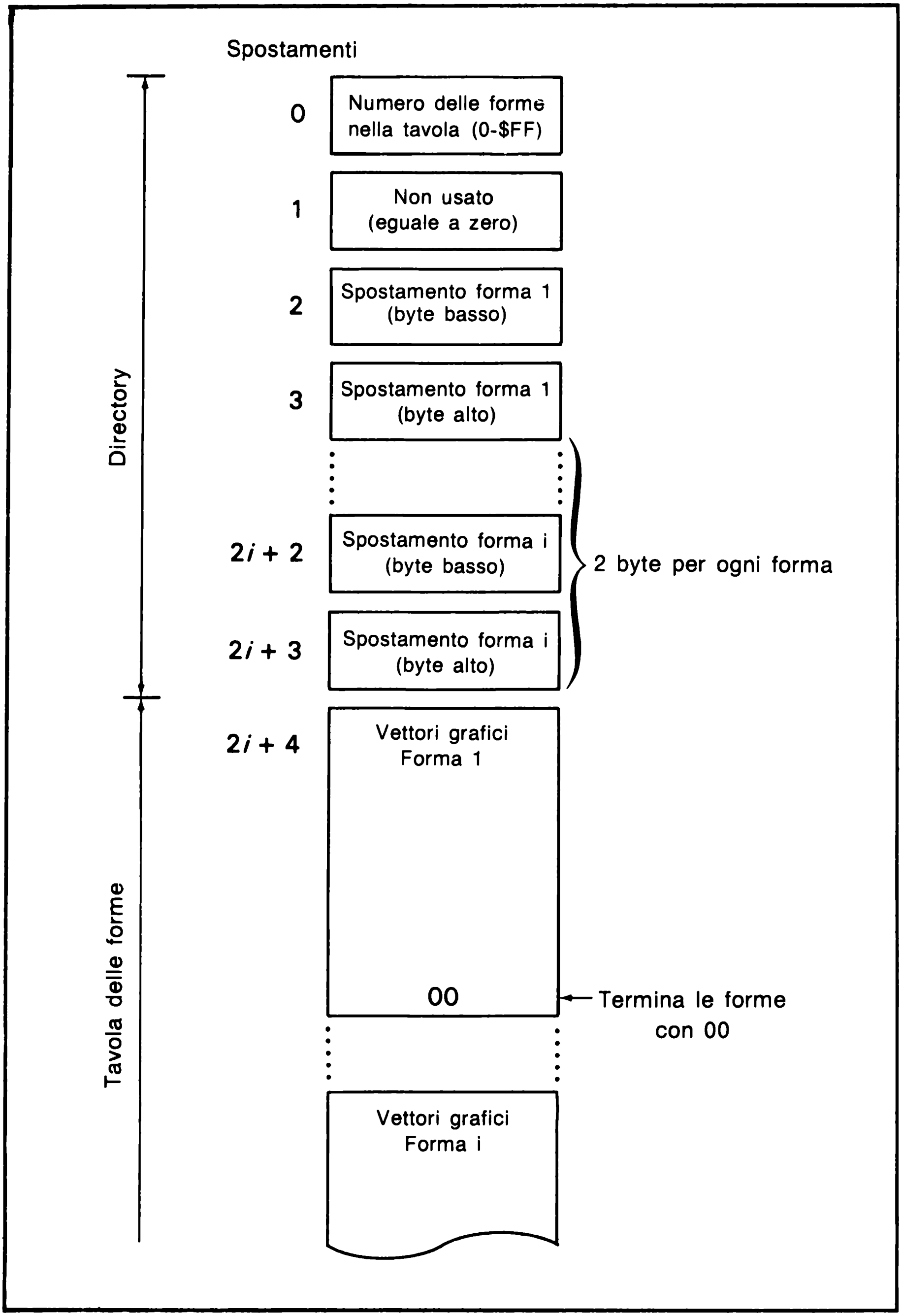


Figura 6.3 – Organizzazione della tavola delle figure.

Tabella 6.5 — Esempio di tavola delle figure (forma quadrata).

	Vettori			Binario			Esadecimale
	Sec. 1	Sec. 2	Sec. 3	Sec. 1	Sec. 2	Sec. 3	
Byte 0	Nessuno	↑	→	00	100	101	25
Byte 1	Nessuno	↓	←	00	110	111	37
Byte 2	Nessuno	Nessuno	Nessuno	00	000	000	00
Byte 3	—	—	—	—	—	—	—

Preparazione dei vettori grafici mediante il calcolatore

Il seguente programma in Applesoft prepara la definizione di una figura. Esso chiede in ingresso i vettori grafici e/o ausiliari. Dopo che avete dato l'ultimo vettore battete una E per "end" (fine) e premete RETURN. Dopodichè il programma vi chiede se dovete cambiare qualche vettore, nel caso che abbiate commesso qualche errore. Se dovete cambiarne qualcuno battete il suo numero e poi il vettore corretto.

Al termine delle correzioni, o se non dovete farne alcuna, battete 0 dopo la domanda VETTORE DA CAMBIARE (0 equivale a fine). Dopo alcuni secondi vi viene visualizzata la lista dei vettori in esadecimale.

Ecco il programma ed un esempio di esecuzione:

```
1   REM  ********************************
2   REM  *  PROGRAMMA PER LA CREAZIONE  *
3   REM  *  DI UNA TAVOLA DELLE FORME   *
4   REM  ********************************
10  DIM S1(100),V1(100)
20 I = 0
30  PRINT "CREAZIONE TAVOLA DELLE FORME"
40  PRINT
50 V = I: GOSUB 270
58  REM L'INGRESSO CONTINUA SINO A CHE
59  REM M$ EGUAGLIA IL VALORE TERMINALE "E"
60  IF M$ <  > "E" THEN S1(I) = M:I = I + 1: GOTO 50
70  PRINT
71  REM  CORREZIONI
80  INPUT "VETTORE DA CAMBIARE (0=FINE):";V
90  IF V > 0 THEN V = V - 1: GOSUB 270:S1(V) = M: GOTO 80
99  REM VETTORI IMPACCATI IN V1()
100  FOR V = 0 TO I
110  IF B = 2 AND S1(V) > 0 AND S1(V) < 4 THEN 140
120  IF B < 2 AND (S1(V) > 0 OR S1(V) > 4) THEN 140
130 B = 0:Q = Q + 1
140 V1(Q) = V1(Q) + S1(V) * (8 ^ B)
150 B = B + 1
160  IF B > 2 THEN B = 0:Q = Q + 1
170  NEXT V
178  REM VISUALIZZAZIONE DEI VETTORI
```

```
179  REM COME NUMERI ESADECIMALI
180  PRINT "BYTE","VETTORE"
190  FOR V = 0 TO Q
200 H% = V1(V) / 16
210 L% = V1(V) - H% * 16
220  IF H% > 10 THEN H% = H% + 7
230  IF L% > 10 THEN L% = L% + 7
240  PRINT V, CHR$ (H% + 176); CHR$ (L% + 176)
250  NEXT V
260  END
269  REM SUBROUTINE INGRESSO VETTORI
270  PRINT "VETTORE ";V + 1;":";
280  INPUT "MUOVI:SU/GIU/SIN/DES?";M$
290 M = 0
300  IF M$ = "DES" THEN M = 1
310  IF M$ = "GIU" THEN M = 2
320  IF M$ = "SIN" THEN M = 3
330  IF M$ = "E" THEN  RETURN
340  INPUT "TRACCIA (Y=SI,N=NO)? ";P$
350  IF P$ = "Y" THEN M = M + 4: RETURN
360  IF P$ = "N" THEN  RETURN
370  GOTO 340

]RUN
CREAZIONE TAVOLA DELLE FORME

VETTORE 1:MUOVI:SU/GIU/SIN/DES?DES
TRACCIA (Y=SI,N=NO)? Y
VETTORE 2:MUOVI:SU/GIU/SIN/DES?DES
TRACCIA (Y=SI,N=NO)? Y
VETTORE 3:MUOVI:SU/GIU/SIN/DES?SU
TRACCIA (Y=SI,N=NO)? N
VETTORE 4:MUOVI:SU/GIU/SIN/DES?SU
TRACCIA (Y=SI,N=NO)? N
VETTORE 5:MUOVI:SU/GIU/SIN/DES?GIU
TRACCIA (Y=SI,N=NO)? N
VETTORE 6:MUOVI:SU/GIU/SIN/DES?GIU
TRACCIA (Y=SI,N=NO)? N
VETTORE 7:MUOVI:SU/GIU/SIN/DES?DES
TRACCIA (Y=SI,N=NO)? Y
VETTORE 8:MUOVI:SU/GIU/SIN/DES?DES
TRACCIA (Y=SI,N=NO)? Y
VETTORE 9:MUOVI:SU/GIU/SIN/DES?SU
TRACCIA (Y=SI,N=NO)? N
VETTORE 10:MUOVI:SU/GIU/SIN/DES?E

VETTORE DA CAMBIARE (0=FINE):0
BYTE            VETTORE
0               2D
1               00
2               90
3               2D
4               00
5               00
```

CARICAMENTO DELLE TAVOLE DELLE FIGURE

Prima di poterle usare, le figure devono essere caricate in memoria. È quindi necessario stabilire in quale area di memoria farle risiedere.

Il modo più semplice per fare questo è quello di posizionare il puntatore HIMEM: ad un valore appena inferiore a quello di inizio del DOS, o appena prima della pagina grafica ad alta risoluzione che intendete usare. Ricordatevi però che dovete posizionare HIMEM: *prima* di eseguire qualunque istruzione in Applesoft che usi delle stringhe. Se usate l'Applesoft nella versione disco, dovete avere almeno 36K di memoria (sebbene sia preferibile averne 48K) per fare coabitare sia l'interprete Applesoft che il DOS. Gli indirizzi 115 e 116 ($73 e $74) contengono il valore più recente di HIMEM: per l'Applesoft (il byte di ordine inferiore è memorizzato per primo).

Per calcolare il nuovo valore di HIMEM:, che permette la memorizzazione di una tavola delle figure, usate la seguente istruzione:

```
PRINT PEEK (116) * 256 + PEEK (115) – X
```

Il parametro *X* rappresenta la lunghezza della tavola compresa la sua directory. Pontete allora il valore di HIMEM:, così calcolato, in memoria prima di caricare la tavola; la tavola risulterà allora protetta da eventuali sconfinamenti dell'Applesoft.

Un'altra zona di memoria in cui potete inserire una tavola delle figure è quella compresa tra gli indirizzi 768 e 975 ($300 e $3CF) inclusi. Fate però attenzione che non vi sia sovrapposizione con qualche subroutine in linguaggio macchina posta in quell'area da voi o dall'Applesoft.

Per caricare in memoria una tavola potete usare le istruzioni POKE. Per esempio le istruzioni seguenti caricano la tavola del quadrato a partire dall'indirizzo 768:

```
]POKE 768,01

]POKE 769,00

]POKE 770,04

]POKE 771,00

]POKE 772,37

]POKE 773,55

]POKE 774,00
```

Una tavola delle figure può anche essere caricata mediante il Monitor. Date quindi il comando CALL-151 per richiamare il Monitor. Battete poi l'indirizzo di memoria in *esadecimale* da cui fare iniziare la tavola seguito da due punti. Continuate ancora e battete il primo byte della directory della tavola seguito da uno spazio vuoto, poi bat-

tete il secondo byte e uno spazio vuoto e così via per tutta la directory. Alla fine date RETURN. Battete ora altri due punti seguiti dai byte in esadecimale della prima figura (i byte sono sempre separati tra di loro da spazi vuoti) e alla fine date ancora RETURN. Ripetete quest'ultimo passo per ogni figura della tavola. Potete controllare quanto caricato battendo l'indirizzo iniziale in esadecimale della tavola, un punto, l'indirizzo finale e RETURN. Per una descrizione dettagliata sull'uso del Monitor vi rimandiamo al capitolo 7. Ecco come dovreste caricare la tavola del quadrato:

`]CALL -151`	
`*`	Segno del Monitor
`*6000:01 00 04 00`	Entra la directory
`*:2C 3E 00`	Entra la forma 1
`*6000.6006`	Controllo con lettura della memoria
`6000- 01 00 04 00 2C 3E 00`	
`*`	

La tavola è ora in memoria. Il primo dato ($6000), che abbiamo fornito, indica l'indirizzo iniziale della tavola. I due punti (:) dicono al Monitor di porre in memoria i valori esadecimali. Subito dopo i due punti viene la directory: 01 (il numero di figure nella tavola), 00 (il secondo byte che non è mai usato) e 04 00(lo spostamento della figura 1 dall'inizio della directory con il byte di ordine più basso posto per primo). La linea successiva inizia con due punti senza che sia necessario ripetere l'indirizzo di caricamento perchè il Monitor pone i nuovi dati in successione ai precedenti.

L'ultima linea dice al Monitor di visualizzare le posizioni di memoria dall'indirizzo $6000 al $6006. Il formato per questo comando è: indirizzo iniziale, un punto (che costituisce il comando di visualizzazione) e l'indirizzo finale.

Memorizzazione delle tavole di figure su cassetta o su disco

Costruire le tavole delle figure richiede molto tempo e spesso è un lavoro tedioso. Per questo motivo è opportuno memorizzarle, appena possibile, su una memoria esterna a cassette o a disco. Come per la maggior parte dei file, anche in questo caso è preferibile usare i dischi per la loro maggior facilità di accesso.

L'Applesoft ha uno speciale comando, SHLOAD, per memorizzare le tavole su cassetta. Prima di memorizzarne una è necessario però calcolarne la lunghezza in esadecimale. Se prendiamo come riferimento la nostra tavola del quadrato vediamo che la sua lunghezza è di 7 byte. Carichiamo quindi questo valore con il Monitor all'indirizzo 0:

```
*00:07 00
```

Ricordiamo che il byte di ordine basso deve sempre precedere l'altro.

Riavvolgiamo la cassetta e poniamo il registratore in RECORD. Useremo adesso il Monitor per registrare la lunghezza della tavola e la tavola stessa mediante due comandi di scrittura posti su una linea:

```
*0.1W 6000.6006W
```

Il primo comando di scrittura trasferisce sulla cassetta i due byte della lunghezza della tavola (indirizzi di memoria 0 e 1). Il secondo comando trasferisce invece i sette byte di tutta la tavola dall'indirizzo \$6000 a quello \$6006. Queste due registrazioni richiedono circa 20 secondi. Udirete due "beep"; dopo il secondo fermate il registratore. La tavola del quadrato è ora registrata su cassetta.

Per portare su disco una tavola delle figure dovete richiamare il DOS (dal Monitor date il comando 3DOG). Il comando DOS che usiamo è BSAVE seguito dall'indirizzo di partenza della tavola e dalla sua lunghezza. Nel caso della tavola del quadrato l'indirizzo è \$6000 e la lunghezza è 7. Ecco il comando:

```
BSAVE SQUARE, A$6000, L7
```

Esso crea il file SQUARE (QUADRATO) di tipo B che contiene la tavola del quadrato in forma binaria.

Caricamento delle tavole di figure da cassetta o da disco

Se avete memorizzato una tavola delle figure su cassetta, per rileggerla in Applesoft dovete dare il comando:

```
SHLOAD
```

Premete il tasto PLAY del registratore. Se tutto funziona correttamente udirete due "beep". Diversamente potrete vedere sullo schermo dei messaggi di errore ERR; in tal caso provate a ripetere il caricamento o anche a controllare il volume del registratore come avevamo indicato nel capitolo 2.

Il comando SHLOAD pone automaticamente HIMEM: al suo valore corrente meno la lunghezza della tavola in byte. Fate quindi attenzione che il valore, che avete per HIMEM: prima di dare SHLOAD, sia corretto.

Se caricate invece la tavola da dischetto dovete posizionare voi HIMEM: prima di caricarla. Il comando di caricamento da dischetto è il seguente:

```
BLOAD SQUARE
```

Il DOS ricorda l'indirizzo iniziale di caricamento ($6000 nel nostro esempio), ma è possibile porre la tavola da un altro indirizzo (per esempio da $3000) in questo modo:

```
BLOAD SQUARE, A$3000
```

Dopo che avete caricato una tavola di figure con BLOAD dovete porre il suo indirizzo nelle posizioni 232 e 233 ($E8 e $E9) che funzionano come puntatore iniziale della tavola in Applesoft. Il comando SHLOAD pone invece questo puntatore automaticamente. Dal Monitor carichiamo il nostro indirizzo $6000:

```
E8:00 60
```

Diversamente potete usare l'istruzione POKE per inserire l'indirizzo di partenza della tavola nelle posizioni 232 e 233, ma ricordatevi che in questo caso dovete usare valori *decimali.*

Siamo ora finalmente in grado di usare le tavole delle figure nei programmi in Applesoft.

COMANDI PER L'USO DELLE FIGURE

In Applesoft esistono quattro istruzioni che permettono di tracciare, cancellare e cambiare orientazione delle figure che abbiamo in precedenza definito:

DRAW	Visualizza la figura sullo schermo
XDRAW	Cancella la figura
ROT	Ruota la figura
SCALE	Varia l'ampiezza della figura

Queste istruzioni fanno uso della pagina grafica ad alta risoluzione che avete in precedenza selezionata con HGR o HGR2 nel colore scelto con HCOLOR.

Il comando SCALE

Questa istruzione deve sempre comparire *prima* di tracciare una figura per la prima volta in un programma (sia in modo differito che immediato).

```
SCALE=1
```

In questo caso significa che i singoli vettori grafici sono tracciati una sola volta. Se avessimo posto SCALE = 5 ognuno di essi sarebbe stato ripetuto 5 volte ottenendo così un disegno 5 volte più grande. Il massimo valore del parametro è 255 che corrisponde ad una figura 255 volte più grande. Attenzione però che è possibile dare anche il valore SCALE = 0 che significa ampliare la figura di 256 volte!

Il comando DRAW

DRAW traccia la figura con numero tra 1 e 255 della tavola presente in memoria. Essa fa riferimento all'ultimo colore scelto e ai fattori di scala e di rotazione già determinati. L'istruzione:

```
DRAW 1 AT 140,96
```

traccia la figura 1 a partire dal punto dello schermo di coordinate 140,96. Se le coordinate del punto iniziale non sono poste l'istruzione DRAW traccia la figura a partire dall'ultimo punto richiamato da una precedente HPLOT o DRAW. Per esempio:

```
DRAW 11
```

In tal modo è possibile tracciare con continuità più figure. Se tale punto non era però già stato definito, vien imposto allora il valore di default 0,0.

IMPORTANTE: l'Applesoft presume che la tavola delle figure sia correttamente caricata in memoria. Per questo motivo, prima di iniziare a tracciare un grafico, controllate che la tavola sia presente e che il suo puntatore sia posizionato agli indirizzi 232 e 233 ($E8 e $E9). Se richiamate un numero di figura che non esiste, o se date delle coordinate non valide, allora il grafico non viene tracciato e appare il messaggio di errore? ILLEGAL QUANTITY ERROR.

Il comando XDRAW

Con XDRAW potete cancellare una figura che sia già presente sullo schermo senza modificare il resto del grafico. Ecco un esempio:

```
XDRAW 8 AT 90,96
```

Questa istruzione è praticamente identica a DRAW; le coordinate del punto iniziale possono essere esplicite oppure implicite come nell'ultimo esempio di DRAW. XDRAW controlla il colore della figura da cancellare poi la traccia nuovamente con il colore *complementare*. Nella Tabella 6.6 trovate i colori complementari di quelli in alta risoluzione.

Tabella 6.6 — Colori complementari in alta risoluzione.

Colori complementari	
Nero	Bianco
Bianco	Nero
Violetto	Verde
Arancione	Blu
Verde	Violetto
Blu	Arancione

In questo modo la figura viene praticamente cancellata perchè i *due colori si compensano;* si comprende anche che il resto del grafico non viene modificato.

Il comando ROT

Questo comando fa ruotare la figura sul piano dello schermo. Per esempio:

```
ROT=16
```

fa ruotare di 90 gradi in senso orario. I valori possibili per l'argomento di ROT sono compresi tra 0 e 255, sebbene siano consentite solo 64 possibili rotazioni tra 0 e 63. In Figura 6.4 sono rappresentati alcuni esempi di rotazioni.

Se il fattore di scala SCALE è posto eguale a 1, ROT ruota la figura solo per multipli di 90°. Ciò significa che solo alcuni valori di rotazione sono validi: 0 = 0°, 16 = 90°, 32 = 180° e 48 = 270°. Per i valori intermedi l'Applesoft approssima al multiplo di 90° immediatamente inferiore. Se invece SCALE è posto eguale o superiore a 5 sono possibili tutte le 64 diverse rotazioni.

Uso delle figure in un programma

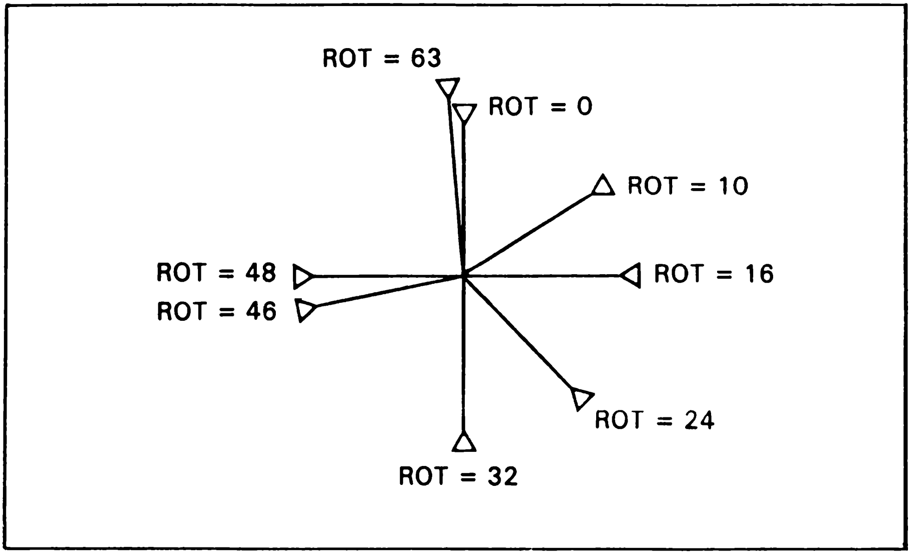
Il seguente programma è un buon esempio di come usare le figure:

```
1  LOMEM: 24600
20  HGR2
50  REM  SET SHAPE TABLE START ADDRESS
60  POKE 232,0
70  POKE 233,96
75  REM  USE ALL COLORS IN TURN
76  FOR H = 1 TO 7
79  REM  INCREMENT ROTATION FACTOR
80  FOR I = 1 TO 80
82  HCOLOR= H
90  ROT= I
92  IF I < 50 THEN  SCALE= I
105  DRAW 1 AT 140,96
106  ROT= I +' 32
110  DRAW 1 AT 140,96
130  NEXT I
140  NEXT H
150  GOTO 76
```

Lavorare con le figure in alta risoluzione è particolarmente utile nella programmazione di giochi.

È importante fare notare che il loro uso è molto più vantaggioso rispetto a quello delle istruzioni HPLOT. Anche l'occupazione di memoria è abbastanza più ridotta e sono necessari meno calcoli, rispetto a quelli che si devono fare con HPLOT. Forse all'inizio troverete qualche difficoltà a preparare le tavole di figure, ma poi vi accorge-

Figura 6.4 — Rotazione delle forme.



rete quanto siano comode. Ricordatevi poi che potete sempre usare quel programma, che vi abbiamo già presentato, per la preparazione automatica dei codici dei vettori grafici.

PARTE SONORA DEL CALCOLATORE APPLE II

Dopo quanto visto sinora, per la programmazione dei grafici, sembrerà molto facile parlare della parte sonora dell'Apple II. Infatti programmare l'altoparlante dell'Apple II è molto semplice, ma contemporaneamente può essere tedioso perchè non esistono comandi BASIC del tipo "sonoro".

Ogni suono che il vostro calcolatore produrrà dovrà essere da voi definito in tutti i dettagli. In pratica la parte sonora dell'Apple II può solamente produrre dei *brevi suoni.* Il trucco per ottenere qualcosa di musicale è quello di variare la frequenza di questi suoni e porli in successione con un certo ritmo.

Vediamo ora come fare tutto ciò in pratica.

FUNZIONAMENTO DEL SONORO

L'Apple II usa la posizione di memoria −16336 (indicata anche come 49200 o $CO30), come se fosse un *tasto,* in modo analogo agli interruttori grafici che abbiamo visto in precedenza.

Ogni volta che accedete a questa posizione di memoria il calcolatore genera un suono. In BASIC potete operare con queste istruzioni:

```
A=PEEK(49200)
```

oppure:

```
A=PEEK(-16336)
```

Se ora pensate di usare un programma BASIC per generare un suono continuo, vi accorgerete che potete produrre solo frequenze molto basse perchè il BASIC è piuttosto lento nella sua esecuzione. In Integer BASIC la massima frequenza è di 256 Hz (cicli al secondo), mentre in Applesoft è di soli 72 Hz. L'unico modo per generare frequenze superiori è quello di usare delle routine in linguaggio macchina.

```
 9 REM  CLICK THE SPEAKER
10 A= PEEK (-16336)
20 GOTO 10
```

Subroutine sonora in linguaggio macchina

Sia l'Integer BASIC che l'Applesoft lasciano libera la zona di memoria, tra gli indirizzi 768 e 975 ($300 e $3CF), che può essere occupata da nuove subroutine senza dovere spostare il puntatore LOMEM:. Attenzione però che il DOS usa quest'area, cancellandone il suo contenuto precedente, quando fate il suo booting.

Per caricare questa subroutine, richiamate il Monitor mediante il comando CALL -151 oppure premendo RESET. (*Attenzione:* se avete il DOS assicuratevi di averlo caricato prima di richiamare il Monitor). Caricate quindi la subroutine sonora in questo modo:

```
>CALL -151

*F666G

!302:LDY 301

0302-   AC 01 03      LDY   $0301
! LDX 301

0305-   AE 01 03      LDX   $0301
! LDA #4

0308-   A9 04         LDA   #$04
! JSR FCA8

030A-   20 A8 FC      JSR   $FCA8
! LDA C030
```

Il calcolatore fa apparire le ombreggiature sopra ai vostri dati

```
030D-    AD 30 C0      LDA    $C030
! INX

0310-    E8            INX
! BNE 310

0311-    D0 FD         BNE    $0310
! DEY

0313-    88            DEY
! BNE 305

0314-    D0 EF         BNE    $0305
! DEC 300

0316-    CE 00 03      DEC    $0300
! BNE 302

0319-    D0 E7         BNE    $0302
! RTS

031B-    60            RTS
```

Gli indirizzi $300 e $301 sono usati come memoria per i dati, mentre quelli da $302 e $31B contengono la subroutine. Dopo il caricamento della subroutine controllatene l'esattezza in questo modo:

```
!$FF69G

*302L

0302-    AC 01 03      LDY    $0301
0305-    AE 01 03      LDX    $0301
0308-    A9 04         LDA    #$04
030A-    20 A8 FC      JSR    $FCA8
030D-    AD 30 C0      LDA    $C030
0310-    E8            INX
0311-    D0 FD         BNE    $0310
0313-    88            DEY
0314-    D0 EF         BNE    $0305
0316-    CE 00 03      DEC    $0300
0319-    D0 E7         BNE    $0302
031B-    60            RTS
031C-    20 20 70      JSR    $7020
031F-    08            PHP
0320-    18            CLC
0321-    D8            CLD
0322-    88            DEY
0323-    08            PHP
0324-    A0 A0         LDY    #$A0
0326-    10 38         BPL    $0360
* <---- Dare CTRL-B
```

Se avete il DOS potete portare su dischetto questa subroutine con il comando BSAVE:

```
BSAVE SOUND,A$302,L26
```

Esso crea il file binario di nome SOUND.

Se avete invece un registratore a cassette, memorizzatela su cassetta con il comando del Monitor:

```
*302.31BW
```

La subroutine può quindi essere usata in futuro.

Collegamento con il BASIC

Per lavorare in BASIC, con la precedente subroutine, è necessario scriverne un'altra di collegamento in Applesoft o in Integer BASIC:

```
3200  REM  SPEAKER DRIVER
3210  POKE 768,D
3220  POKE 769,F
3230  CALL 770
3240  RETURN
```

Con queste due subroutine, l'ultima in BASIC richiama la prima in linguaggio macchina, potete scrivere i vostri programmi sonori in BASIC definendo i parametri F (frequenza, compreso tra 1 e 255) e D (durata, compreso anche tra 1 e 255). Per prova eseguite il seguente programma:

```
10  FOR I = 1 TO 254
20 F = 1
30 D = I
40  GOSUB 3200
50  NEXT I
60  END
```

Durante l'esecuzione fate attenzione alle note. Quelle estreme a bassa e alta frequenza sono più corte di quelle a frequenza media. Questo è dovuto al fatto che il programma usa dei cicli e non può fare uso di tempi reali perchè manca l'orologio nel calcolatore. Voi potete però compensare queste diversità imponendo tempi di durata D diversi tra le note estreme e quelle intermedie.

Un programma sonoro più complesso

Il programma listato qui sotto usa le subroutine sonore già descritte per creare una

serie di suoni che voi potete ascoltare, modificare e stampare per usarli in altri programmi BASIC.

Durante l'esecuzione del programma vi appare il messaggio (E)NTER, (L)ISTEN, (P)RINT? (E per ingresso, L per ascolto e P per stampa) per chiedervi che cosa fare. All'inizio battete ovviamente E per impostare il primo tono e premete RETURN.

Vi appare allora TONE 1: FREQUENCY? DURATION? (TONO 1: FREQUENZA, DURATA?) per chiedervi quale frequenza e quale durata deve avere il tono 1. Battete due numeri, compresi tra 1 e 255, separati da una virgola; appena premete RETURN il calcolatore produce il tono 1. Questo processo si ripete varie volte sino al massimo di 100 toni. Per terminare il caricamento date 0 per il tono e 0 per la durata.

Se ora volete cambiare qualche tono potete rispondere con il numero del tono alla domanda WHICH NOTE TO CHANGE? (QUALE NOTA CAMBIARE?). Se non ne volete cambiare date il valore 0.

Dopo i cambiamenti ritorna la domanda (E)NTER, (L)ISTEN, (P)RINT?. Battete quindi L e RETURN per ascoltare tutte le note assieme. Alla fine riappare il messaggio di chiamata e voi potete dare il comando di stampa P. Sullo schermo vi appare così l'elenco di tutte le note con la loro frequenza e durata.

Provate da voi ad aggiungere le istruzioni per salvare, su cassetta o dischetto, i pezzi musicali che avete scritto.

```
10  REM  SOUND GENERATOR PROGRAM
19  REM  ARRAY REMEBERS ENTERED TONES
20  DIM A(100,2)
29  REM  CLEAR DISPLAY
30  CALL  - 936
40  INPUT "(E)NTER, (L)ISTEN, (P)RINT?";A$
50  IF A$ = "L" THEN 1000
60  IF A$ = "P" THEN 1200
80  IF A$ <  > "E" THEN 30
81  REM  ENTER EACH TONE
90  PRINT
100 I = 0
105 M = I
110  GOSUB 3000
119  REM  END OF TONE ENTRY?
120  IF F = 0 AND D = 0 THEN I = I - 1: GOTO 200
129  REM  NO--REMEMBER TONE
130 A(I,1) = F:A(I,2) = D
140 I = I + 1
150  GOTO 105
200  REM  CHANGE ANY ITEMS HERE
205  PRINT "WHICH NOTE TO CHANGE (0-";I;")";
206  INPUT E
208  IF E = 0 THEN 30
210  IF E < 1 OR E > I THEN 210
220 M = E: GOSUB 3000
230 A(E,1) = F:A(E,2) = D: GOTO 205
1000  REM  LISTEN TO THE NOTES SO FAR
```

```
1010  FOR K = 0 TO I
1020 F = A(K,1):D = A(K,2): GOSUB 3200
1030  NEXT K
1040  GOTO 30
1200  REM  PRINT OUT THE NOTES
1210  PRINT "NOTE#","FREQ","DURATION"
1220  FOR K = 0 TO I
1230  PRINT K,A(K,1),A(K,2)
1240  NEXT K
1250  PRINT
1255  PRINT "PRESS RETURN TO CONTINUE": INPUT Z$
1260  GOTO 30
3000  PRINT "TONE ";M;
3010  INPUT " ENTER FREQUENCY, DURATION";F,D
3015  IF F = 0 AND D = 0 THEN  RETURN
3020  IF (F < 0 OR F > 255) OR (D < 1 OR D > 255) THEN 3010
3030  GOSUB 3200
3040  RETURN
3200  REM  SPEAKER DRIVER
3210  POKE 768,D
3220  POKE 769,F
3230  CALL 770
3240  RETURN
```

CAPITOLO 7

PROGRAMMA DI CONTROLLO MONITOR IN LINGUAGGIO MACCHINA

Il programma di controllo Monitor risiede permanentemente nella memoria di sola lettura ROM dell'Apple II. In questo capitolo descriviamo le caratteristiche del Monitor e di come usarlo assieme ai vostri programmi BASIC.

Il Monitor, che è scritto in linguaggio macchina, ha lo scopo di costituire un collegamento (link) tra il BASIC (o gli altri linguaggi superiori forniti con il calcolatore Apple II) e le varie funzioni a basso livello eseguite dalla macchina, come stampare un carattere, tracciare una linea, ecc.

È possibile anche lavorare direttamente con il Monitor, tramite la tastiera, come abbiamo già visto fare per creare le tavole di figure grafiche (capitolo 6) o per analizzare il contenuto della memoria alla ricerca di possibili guasti dell'hardware o per scrivere un qualunque programma in linguaggio macchina. Non pensiate però che sia fondamentale conoscere bene il Monitor, ma una sua conoscenza anche approssimata è senz'altro molto utile.

In questo capitolo vi diamo una descrizione del Monitor e del Mini-Assembler, ma non desideriamo entrare nei dettagli della programmazione vera e propria in assembler. Se desiderate approfondire questo tipo di programmazione potete consultare uno dei testi che abbiamo riportato nell'Appendice K.

COME ACCEDERE AL MONITOR

Esistono due versioni di Monitor: quella standard e quella Autostart.

Se il vostro calcolatore ha il Monitor standard, quando lo accendete siete subito sotto il suo controllo; lo schermo vi apparirà all'accensione pieno di caratteri vari con l'asterisco (*) e il cursore nell'angolo in basso a sinistra. L'asterisco è appunto il carattere di pronto del Monitor.

Se avete invece l'Austostart Monitor (come opzione della vostra macchina o come parte standard di un Apple II Plus) dovete passare tramite l'Integer BASIC, o l'Apple-

soft, per accedere al Monitor. Dopo il carattere di pronto del basic (> per l'Integer BASIC o] per l'Applesoft), battete il seguente comando:

```
CALL -151
```

Questa istruzione è in effetti una chiamata alla subroutine con indirizzo di partenza FF69; il BASIC non riconosce però i parametri esadecimali per cui dovete dare il valore equivalente in decimale. Subito dopo questo comando vi appariranno il carattere di pronto del Monitor e l'asterisco.

RITORNO DAL MONITOR

È possibile uscire dal Monitor in diversi modi a seconda di quello che desiderate fare ritornando in BASIC.

Se volete salvare un programma BASIC e le sue variabili, premete allora CTRL-C e RETURN. Dopo RETURN vi apparirà il carattere di pronto del BASIC. Se volete potete allora listare il programma o stampare il valore delle sue variabili.

Per illustrare questo supponiamo che abbiate posto un certo valore in una posizione di memoria mediante una POKE e vogliate controllare tale operazione. Ovviamente potreste usare una istruzione PEEK, ma noi vogliamo farvi vedere come sia preferibile usare il Monitor. Il programma seguente vi mostra come, usando CTRL-C, si mantiene il programma BASIC e le sue variabili ritornando dal Monitor:

```
>10 A=123
>19 REM MOVE CURSOR TO 13TH COLUMN
>20 POKE 36,12
>30 PRINT A
>40 END
>RUN
            123

>CALL -151

* <---- Premere CTRL-C e RETURN

>PRINT A
123

>LIST
   10 A=123
   19 REM  MOVE CURSOR TO 13TH COLUMN
   20 POKE 36,12
   30 PRINT A
   40 END
>
```

Se volete invece ritornare dal Monitor, cancellando il programma BASIC presente e le sue variabili, premete allora CTRL-B e RETURN. Provate infatti a sostituire CTRL-

C con CTRL-B nell'esempio precedente e vi accorgerete che non è più possibile listare il programma o richiamare le sue variabili.

Attenzione però che CTRL-C opera con l'Integer BASIC e con l'Applesoft su cartolina, ma *non funziona* con l'Applesoft proveniente da cassetta o da disco.

In quest'ultimo caso, cioè se l'Applesoft è nella versione su cassetta o su disco, potete usare CTRL-B per uscire dal Monitor, ma perderete il vostro programma. Per ovviare a questo inconveniente potete usare questo comando:

```
*3D0G
```

nel caso di Applesoft proveniente da disco; oppure quest'altro comando:

```
*0G
```

nel caso di Applesoft proveniente da cassetta.

I due comandi 3D0G e 0G sono analoghi all'istruzione di salto del BASIC GOTO.

IMPORTANTE: potete usare il comando 0G *solamente* con l'Applesoft proveniente da cassetta.

FUNZIONI DEL MONITOR

Il Monitor esegue poche funzioni, ma ognuna di estrema importanza. Apprezzerete ancora di più l'importanza del Monitor se terrete presente che esso è un programma molto breve.

Con il Monitor potete esaminare le singole posizioni di memoria oppure gli stessi registri del microprocessore; potete ricopiare la memoria (dump) sullo schermo o su un'altra periferica d'uscita come una stampante. Potete anche cambiare il contenuto della memoria o dei registri. Altre funzioni previste dal Monitor permettono di muovere interi blocchi di dati da un indirizzo ad un altro o di confrontarli tra loro. Altre funzioni saranno descritte nel seguito di questo capitolo.

ANALISI DELLA MEMORIA

Il Monitor prevede tre metodi per leggere la memoria: *per singolo indirizzo, per parola* e *per blocco.*

Nel primo caso si può leggere un byte di memoria alla volta; nel secondo caso leggere una parola vuol dire leggere otto byte consecutivi a partire da un indirizzo divisibile per otto; nell'ultimo caso si può leggere la memoria da un indirizzo ad un altro superiore.

Analisi per singolo indirizzo

Per leggere il contenuto di un singolo byte di memoria è sufficiente che battiate il

suo indirizzo, in esadecimale, subito dopo il carattere di pronto del Monitor e poi premiate RETURN. Ecco un esempio:

```
*FF69
```

Il Monitor risponde visualizzando il contenuto di tale posizione:

```
FF69- A9
*
```

Attenzione che l'indirizzo, che in questo modo avete comunicato al Monitor, é come un puntatore utilizzabile per successive letture della memoria. Se desiderate cambiarlo dovete battere il nuovo valore, sempre in esadecimale, dopo l'asterisco del Monitor; per esempio:

```
*300F
```

significa portare il puntatore nella posizione 300F per visualizzarla.

Analisi per parole

Se dopo avere esaminato la posizione di memoria FF69 volete esaminare quelle successive è sufficiente che premiate ancora RETURN. Ecco un esempio:

```
*FF69

FF69- A9
* ←——— Premere RETURN
 AA 85 33 20 67 FD
*

FF70- 20 C7 FF 20 A7 FF 84 34
*←——— Premere ancora RETURN
```

Dopo il primo RETURN vi appaiono sullo schermo sei byte di memoria. Essi rappresentano il contenuto delle posizioni da FF6A a FF6F. Dopo il secondo RETURN vi appaiono invece gli otto byte che costituiscono una intera parola preceduti dall'indirizzo iniziale (FF70) di tutta la parola. Tutte le parole iniziano con gli indirizzi divisibili per otto. Per questo motivo non era apparso l'indirizzo di parola dopo il primo RETURN. Questo primo RETURN aveva semplicemente visualizzato i restanti byte di una parola dopo l'indirizzamento di un suo byte intermedio.

Analisi per blocchi

Per leggere una porzione di memoria, compresa tra due indirizzi e più lunga di otto byte, bisogna porre questi due indirizzi in esadecimale dopo l'asterisco del Monitor.

L'indirizzo inferiore deve essere posto prima del superiore e separato da questo da un punto. Per esempio:

```
*F800.F83F
```

Il Monitor risponde con il contenuto delle posizion interrogate:

```
F800- 4A 08 20 47 F8 28 A9 0F
F808- 90 02 69 E0 85 2E B1 26
F810- 45 30 25 2E 51 26 91 26
F818- 60 20 00 F8 C4 2C B0 11
F820- C8 20 0E F8 90 F6 69 01
F828- 48 20 00 F8 68 C5 2D 90
F830- F5 60 A0 2F D0 02 A0 27
F838- 84 2D A0 27 A9 00 85 30
*
```

Se nel comando avete posto prima l'indirizzo superiore, vi verrà visualizzato solo il suo contenuto (cioè la lettura della memoria non avviene a ritroso).

Se avete indicato due indirizzi che siano distanziati più di quanto può contenere lo schermo, avrete automaticamente uno scorrimento delle righe verso l'alto (scroll) per fare posto ai nuovi dati. Non è possibile cancellare questa visualizzazione senza premere RESET. Se avete l'opzione Autostart Monitor, potete fermare temporaneamente la visualizzazione premendo CTRL-S. CTRL-S ferma l'uscita verso lo schermo, ma non verso gli altri periferici di uscita, come le stampanti, dandovi così la possibilità di leggere lo schermo a vostro piacimento. Per fare continuare la visualizzazione premete la barra di spazio.

L'esame della memoria a blocchi è un esempio tipico di *dump*. Al termine di una lettura per blocchi il puntatore si porta sul byte successivo a quello finale del blocco.

È possibile usare il valore indicato dal puntatore come primo parametro del comando di lettura per blocchi; il secondo parametro deve invece essere presente e preceduto da un punto. Per esempio al termine dell'esempio precedente il puntatore è posizionato sull'indirizzo F840 (successivo a F83F) per cui possiamo leggere il blocco di memoria da F840 a F880, semplicemente scrivendo:

```
*.F880
```

a cui segue sullo schermo:

```
F840- 20 28 F8 88 10 F6 60 48
F848- 4A 29 03 09 04 85 27 68
F850- 29 18 90 02 69 7F 85 26
F858- 0A 0A 05 26 85 26 60 A5
F860- 30 18 69 03 29 0F 85 30
F868- 0A 0A 0A 0A 05 30 85 30
F870- 60 4A 08 20 47 F8 B1 26
F878- 28 90 04 4A 4A 4A 4A 29
F880- 0F
*
```

ESAME DEI REGISTRI DEL MICROPROCESSORE

Il Monitor vi permette anche di ispezionare il contenuto dei registri del microprocessore. Provate infatti a battere CTRL-E e poi RETURN; otterrete qualcosa del genere:

```
A=CD X=B1 Y=C3 P=B5 S=F0
```

I cinque valori in esadecimale rappresentano il contenuto dei registri: A per l'accumulatore, X e Y per i due registri indice, P per il program counter ed S per lo stack pointer. I valori riportati a destra del segno di eguale rappresentano gli ultimi contenuti dei registri stessi, che rimangono memorizzati dal Monitor sino quando eseguirete un programma in assembler o ritornerete in BASIC. In altre parole il Monitor non influisce sui valori dei registri, ma memorizza il loro più recente contenuto.

MODIFICA DELLA MEMORIA

Modificare il contenuto di memoria, cioè scrivere in essa qualcosa, richiede molta più attenzione che non il caso precedente della sola lettura. Ora è infatti necessario fornire anche i valori che rappresentano il nuovo contenuto di memoria.

Mediante il Monitor è possibile modificare un singolo byte alla volta e anche modificare una successione continua di byte.

Modifica di una singola posizione

Per modificare una singola posizione bisogna innanzitutto posizionare il puntatore del Monitor.

Il comando per posizionare il puntatore è lo stesso già visto precedentemente. Battiamo quindi l'indirizzo, che desideriamo modificare, in esadecimale e poi premiamo RETURN. Per esempio:

```
*1200
```

pone il puntatore all'indirizzo esadecimale 1200.

Il Monitor allora risponde visualizzando il contenuto di questa posizione di memoria:

```
1200- 73
```

Notate che così facendo si ottiene lo stesso risultato che si otterrebbe facendo la sola lettura della memoria: viene visualizzato l'indirizzo (1200) e il suo contenuto (73).

Il passo successivo consiste nel cambiare il contenuto della posizione. Il comando per fare questo è il carattere due punti (:) seguito dal nuovo valore in esadecimale.

Per esempio:

```
*:5F
```

Invece di scrivere in memoria in due tempi è possibile battere il comando su una stessa linea in questo modo:

```
*1200:5F
```

Otteniamo così che il valore (5F) venga scritto nella posizione (1200). Dopo questa operazione il puntatore si incrementa puntando la posizione successiva (1201). Se quindi vogliamo scrivere qualcosa (per esempio 7F) nella posizione 1201, è sufficiente che battiamo:

```
*:7F
```

Dopodichè il puntatore si incrementa (1202) e noi possiamo scrivere un nuovo contenuto anche in questa posizione.
In altra parole è possibile scrivere, in posizioni successive di memoria, indicando esplicitamente solo il primo indirizzo; il Monitor provvede ad incrementare gli indirizzi successivi automaticamente.

Modifica di un blocco di memoria

Per riscrivere un blocco di posizioni consecutive di memoria dovete portare il puntatore sulla posizione iniziale, poi dare il comando di scrittura (:) e quindi indicare i nuovi valori in esadecimale tutti di seguito e tra loro separati da uno spazio. Per esempio se vogliamo riscrivere le otto posizioni dall'indirizzo 1200 al 1207, battiamo questa linea di comando:

```
*1200:00 01 02 03 04 05 06 07
```

Come vedete viene esplicitamente indicato solo il primo indirizzo, perchè il puntatore si sposta automaticamente in corrispondenza ad ogni valore posto nella lista.
Il numero massimo di posizioni che potete riscrivere in questo modo è 83 se nel comando avete esplicitato anche l'indirizzo iniziale; è invece 84 se non indicate il puntatore perchè ritenete che sia già posizionato dove vi interessa.
In ambedue i casi, però, la linea di comando sarà così lunga da dovere andare a capo diverse volte. Ciò comporta delle difficoltà nell'eventualità di correzioni di errori nella linea stessa. L'unica possibilità che avete per correggere un errore è quella di ritornare indietro (backspace) e ribattere tutta la linea dal punto errato.

Controllo delle modifiche di memoria

È molto importante che controlliate quello che avete scritto in memoria prima di utilizzarlo. Per fare questo potete usare uno dei tre metodi che abbiamo visto in precedenza per l'analisi della memoria.

Per eseguire questa lettura dovete ovviamente riportare il puntatore nella posizione iniziale. Per esempio per controllare quello che abbiamo scritto nelle posizioni da 1200 a 1207, battiamo:

```
*1200
```

Il Monitor risponde:

```
1200- 00
*
```

Premendo RETURN vi vengono visualizzate anche le altre sette posizioni che avete riscritto:

```
*  ←—— Premere RETURN
 01 02 03 04 05 06 07
*
```

Se avete modificato più di otto posizioni continuate a premere RETURN finchè vi appare l'ultima posizione modificata.

Potete anche fare la lettura per blocchi con il comando:

```
*1200.1207
```

a cui il Monitor risponde:

```
1200- 00 01 02 03 04 05 06 07
*
```

Correzione di errori

Per correggere eventuali errori che avete riscontrato nell'analisi della memoria, il metodo più semplice che potete seguire è quello di riscrivere le singole posizioni errate. Per esempio se avete riscontrato che la posizione 1204 è errata, è sufficiente che battiate:

```
*1204:04
```

Fate quindi le correzioni che ritenete opportune, ma controllate ancora che tutto ciò che avete inserito in memoria sia esatto, prima di passare alla fase successiva del vostro lavoro.

MODIFICA DEL CONTENUTO DEI REGISTRI DEL MICROPROCESSORE

Il processo di modifica del contenuto dei registri è diverso da quello di modifica delle posizioni di memoria perchè i registri non hanno un loro indirizzo.

Per operare sui registri dovete prima esaminarli con il comando CTRL-E, poi subito dopo dare il comando al Monitor di scrittura (:) seguito da uno a cinque valori in esadecimale separati da uno spazio.

I cinque valori in esadecimale corrispondono in ordine all'accumulatore, al registro indice X, al registro indice Y, al program counter e allo stack pointer. Per cambiare il valore di un registro intermedio dovete ripetere il valore dei registri precedenti, ma potete trascurare di ridare il valore a quelli successivi.

Per esempio supponiamo di dover cambiare il contenuto del program counter P e lasciare gli altri inalterati. Cominciamo con la lettura dei registri con CTRL-E:

```
* ←—— Premere CTRL-E e poi RETURN

 A=CD X=B1 Y=C3 P=B5 S=F0
*
```

Dobbiamo allora mantenere il contenuto dell'accumulatore A, dei due registri indice X e Y, battere il nuovo valore del program counter P e possiamo infine trascurare di ripetere il valore dello stack pointer S. Battiamo quindi il seguente comando:

```
*: CD B1 C3 8A
```

L'unico registro che si può cambiare facilmente è l'accumulatore perchè è il primo della lista.

Ricordatevi che per operare sui registri *dovete* sempre usare il comando CTRL-E, perchè diversamente il Monitor esaminerebbe le posizioni di memoria e non i registri.

Facciamo un altro esempio. Supponiamo di volere cambiare lo stack pointer S dal valore FO a 4B. Cominciamo con il premere CTRL-E e RETURN per esaminare il contenuto dei registri:

```
* ←—— Premere CTRL-E e poi RETURN

 A=FF X=CD Y=81 P=8A S=F0
*
```

Battiamo quindi i primi quattro valori inalterati ed il quinto nuovo:

```
*:FF CD 81 8A 4B
```

Ripetiamo infine CTRL-E per controllare che tutto sia corretto.

TRASFERIMENTO DI BLOCCHI DI MEMORIA SU PERIFERICHE

È possibile ricopiare interi blocchi di memoria centrale su periferiche come cassette o dischi. Potete per esempio portare su cassetta dei grafici ad alta risoluzione (vedi il capitolo 6) oppure dei programmi scritti in assembler. In questo caso, cioè se usate le cassette, potete dare tutti i relativi comandi direttamente sotto il controllo del Monitor. Se invece volete usare i dischetti, dovete lasciare temporaneamente il Monitor, e lavorare con i comandi DOS del BASIC.

Deposito su cassetta di un blocco di memoria

Il Monitor permette di comandare la scrittura su cassetta di una intera area di memoria compresa tra due indirizzi. Il formato di questo comando richiede che diate il primo indirizzo, poi un punto, quindi il secondo indirizzo e infine la lettera W.

Per esempio il comando:

```
*2200.2FFFW
```

ordina al Monitor di trasferire su cassetta il contenuto di memoria dall'indirizzo esadecimale 2200 all'indirizzo esadecimale 2FFF.

Attenzione però che questo comando del Monitor non controlla i dati che invia alla cassetta, nè controlla che la cassetta sia presente e funzionante. Dovete voi stessi controllare in precedenza che il registratore sia collegato correttamente, che la cassetta sia inserita e che ogni altra cosa sia pronta per funzionare nel modo previsto.

Un consiglio pratico è il seguente. Prima di premere RETURN, in coda al comando di scrittura, mettete in registrazione (RECORD) la cassetta e premete RETURN solo quando vedrete il nastro muoversi. (Se siete all'inizio della cassetta attendete qualche secondo per lasciare passare la parte iniziale, trasparente non magnetica, del nastro).

Appena premete RETURN il calcolatore attende circa dieci secondi prima di inviare i dati. In questa fase vengono cancellate le registrazioni precedenti e viene contemporaneamente registrato un *segnale di riferimento* che sarà poi utilizzato nella fase di lettura; dopodichè inizia la registrazione dei dati contenuti nella memoria.

Al termine della registrazione riappare sullo schermo il carattere di pronto del Monitor e viene suonato un "beep".

Questo comando di registrazione vi permette di ricopiare da uno a 64K byte (65536 byte) su una cassetta. La velocità di trasferimento dei dati è di circa 210 caratteri al secondo. A questa velocità 16384 byte sono registrati in 77.5 secondi, escluso il tempo di registrazione del segnale di riferimento. Al termine dei dati di memoria viene registrato il byte di *checksum*. Questo byte di controllo sarà poi utilizzato in lettura per verificare la validità dei dati.

Lettura da cassetta

L'operazione inversa della precedente è quella che permette di riportare in memoria un blocco di dati precedentemente registrato su cassetta. Il formato del comando è analogo al precedente salvo che in coda ad esso dovete porre una R invece di una W. Il primo indirizzo rappresenta quello da cui il Monitor inizierà a caricare la memoria, mentre il secondo rappresenta l'indirizzo finale. I due indirizzi devono essere separati da un punto. Ecco un esempio:

```
*2000.20FFR
```

Il comando significa leggere da cassetta e scrivere in memoria dalla posizione 2000 alla posizione 20FF.

A differenza del comando di scrittura, che opera sempre anche con il registratore fermo, il comando di lettura attende che voi abbiate premuto il tasto PLAY del registratore; il calcolatore aspetta poi di ricevere il segnale di riferimento. Prima di premere PLAY assicuratevi che il nastro della cassetta sia posizionato dove inizia il segnale di riferimento. Per fare questo potete staccare il registratore dal calcolatore e collegarlo momentaneamente ad un altoparlante; il segnale di riferimento ha un suono medio e uniforme mentre i dati producono un suono rumoroso e irregolare.

Come già altre volte vi abbiamo detto, prima di lavorare con il registratore a cassette controllate il suo livello di volume con la procedura descritta nel capitolo 2.

Dobbiamo infine porre in risalto il fatto che il Monitor si aspetta di leggere blocchi di dati lunghi esattamente quanto erano quelli registrati. In altre parole non potete leggere meno dati di quanti ne avete in precedenza scritti, nè tentare di leggerne di più. In ambedue i casi è possibile che vi venga segnalato un errore.

Condizioni di errore nel comando di lettura della memoria

Il Monitor prevede di ascoltare per almeno 3.5 secondi il segnale di riferimento che gli proviene dalla cassetta nella fase di lettura per potere fare una sua regolazione elettronica interna. Se questo segnale è più corto di 3.5 secondi il Monitor trascura la prima parte dei dati, per cui si avrà, oltre alla perdita dei dati stessi, anche un errore di checksum. I dati che entrano in memoria non sono a loro volta utilizzabili perchè non possiamo sapere da quale punto il Monitor ha iniziato a leggere correttamente il nastro. Avremo quindi sicuramente una segnalazione di errore sullo schermo e un "beep" sonoro di avviso.

Prima di provare a rileggere i dati dalla cassetta dovete capire se sul nastro è stato registrato il segnale di riferimento sufficientemente lungo. Riavvolgete la cassetta, collegate il registratore ad un altoparlante, premete PLAY e cercate di misurare con un orologio quanto dura il segnale di riferimento. Se esso dura meno di 3.5 secondi allora dovrete sicuramente ripetere la registrazione sulla cassetta. Attenzione: questo inconveniente succede spesso quando si riavvolge completamente una cassetta

e le prime registrazioni vanno quindi a finire sulla parte iniziale non magnetica del nastro.

È opportuno che diciamo qualche parola in più sul byte di checksum che viene posto in coda ai dati sulla cassetta.

Quando siamo in registrazione, il calcolatore prepara un byte di controllo che è funzione di tutti i byte che scriviamo. (Checksum significa appunto somma di controllo). Alla fine questo byte viene registrato anche lui sul nastro.

Quando siamo invece nella fase di lettura, il calcolatore mano mano che legge i byte ricalcola il byte di checksum e poi alla fine lo confronta con quello che era stato registrato. Se i due valori sono eguali, tutto va bene; se sono diversi viene segnalato lo stato di errore perchè sicuramente i dati letti non corrispondono a quelli registrati. È molto poco probabile che più errori si compensino tra di loro così da non dare luogo ad un errore di checksum.

Si comprende quindi che se la parte iniziale della registrazione va persa il calcolatore, non potendo più sapere quale è il vero byte di checksum registrato, farà un controllo di checksum che porterà sicuramente allo stato di errore.

Come regola generale vi raccomandiamo di leggere blocchi di lunghezza sempre eguale a quella usata in scrittura. In questo caso la tecnica del checksum vi dà una garanzia quasi completa dell'esattezza di quanto registrato. In seguito vedremo come usare il Monitor per fare delle operazioni di controllo di lettura

Registrazione di blocchi di memoria su disco

La registrazione di blocchi di memoria su disco è molto più facile e sicura di quella su cassetta. In questo caso si deve usare il DOS, con il BASIC, invece del Monitor. Per poter comprendere quanto diremo qui di seguito, è necessario conosciate bene il DOS (capitolo 5) il quale deve già risiedere in memoria (capitolo 2).

Se siete sotto il controllo del Monitor, dovete temporaneamente lasciarlo e passare al BASIC, e al DOS, con il comandi CTRL-B o CTRL-C se avete l'Autostart Monitor, o 3DOG se avete il Monitor standard.

Ecco quindi un esempio di comando per portare un file dalla memoria sul disco:

```
BSAVE SHPTABLE, A$3000, L256, S6, D1, V201
```

Questo comando crea su disco il file SHPTABLE. Il parametro A (address) indica l'indirizzo iniziale 3000 in forma esadecimale; il parametro L (lunghezza) indica la lunghezza del file da copiare, in questo caso decimale. La lunghezza massima può essere di 32767 byte in decimale ($7FFF in esadecimale). Il comando BSAVE permette di usare per A e L sia valori decimali che esadecimali. I valori esadecimali devono essere preceduti dal segno $.

Gli ultimi tre parametri (S6, D1 e V201) sono facoltativi. S indica il numero di slot ed è necessario usarlo solo se avete più di una cartolina controllo per disco ed inoltre se volete usare un drive collegato ad uno slot diverso da quello usato per ultimo (nor-

malmente si usa lo slot 6 standard). D è il numero di drive; anche lui deve essere indicato solo se usate un drive diverso da quello usato in precedenza. Il parametro di volume V deve essere quello riportato nella directory del dischetto che state usando. Se il numero di volume che indicate nella BSAVE è diverso da quello del dischetto, che avete nel drive, vi viene segnalato un messaggio di errore.

Lettura dei dati dal disco

Il comando inverso di BSAVE, per leggere dei dati dal disco, è BLOAD. Ecco un esempio:

```
BLOAD SHPTABLE, A$3000
```

Questo comando carica il file SHPTABLE dal disco e lo pone in memoria a partire dall'indirizzo iniziale esadecimale 3000. Questo indirizzo può anche essere posto in forma decimale.

L'indirizzo iniziale può non essere indicato, nel comando BLOAD, se si vuole caricare il file dallo stesso indirizzo che era stato indicato nel comando BSAVE; esso è quindi necessario solo per caricare i file da posizioni diverse. I parametri L, S, D e V sono facoltativi.

Attenzione! Quando caricate un file in memoria dovete evitare che esso sconfini in zone occupate dal DOS, dall'interprete Applesoft, dalle pagine grafiche oppure dalle variabili del BASIC perchè ciò comporterebbe inevitabilmente di rovinare i vostri programmi o di perdere i dati.

SPOSTAMENTO E CONFRONTO DI BLOCCHI DI MEMORIA

Quando usate i dischi o le cassette per leggere o scrivere blocchi di memoria, il Monitor vi offre la possibilità di effettuare dei controlli molto importanti. La prima possibilità offerta dal Monitor, è quella di ricopiare dei blocchi di memoria in posizioni con indirizzi diversi; la seconda permette di confrontare due blocchi e segnalare eventuali diversità tra di loro. In questo modo potete verificare se le operazioni di lettura o scrittura della memoria sono avvenute senza errori.

Il comando di spostamento di aree di memoria

Questo comando richiede l'indirizzo iniziale dell'area da copiare (*indirizzo iniziale sorgente*), l'indirizzo finale da copiare (*indirizzo finale sorgente*) e l'indirizzo iniziale da cui riportare i dati ricopiati (*indirizzo iniziale destinazione*). Il formato di questo comando vuole anche che si separi l'indirizzzo iniziale destinazione dai due indirizzi sorgente con un segno di minore (<) e si ponga in coda una M per indicare lo spostamento (dall'inglese move). I due indirizzi sorgente devono essere separati da un punto. Ecco un esempio:

```
*1200<2000.2100M
```

Il blocco di memoria compreso tra i due indirizzi esadecimali 2000 e 2100 viene ricopiato a partire dall'indirizzo destinazione 1200. Il nuovo blocco sarà quindi contenuto tra gli indirizzi 1200 e 1300. Siccome gli indirizzi sono in esadecimale, la lunghezza del blocco ricopiato è di 257 byte in decimale (o 101 byte in esadecimale). Il contenuto dei due blocchi è ovviamente perfettamente eguale.

L'indirizzo finale sorgente deve essere maggiore o eguale all'indirizzo iniziale sorgente. Se invece è minore, il Monitor sposterà solamente il primo byte dall'area sorgente.

Riempimento della memoria

Con il comando di spostamento di dati in memoria è possibile *riempire* (*fill*) una zona di memoria con dati tutti uguali; cioè inserire uno stesso dato in maniera ripetitiva in una intera area.

Supponiamo per esempio di volere inserire degli zeri nel blocco di memoria tra gli indirizzi 1D00 e 1DFF.

Per prima cosa poniamo a zero il primo byte del blocco:

```
*1D00:00
```

Mediante il comando di spostamento porteremo ora questo byte nullo in tutte le posizioni del blocco di memoria. Definiamo come indirizzo iniziale di destinazione la prima posizione successiva (1D01). L'indirizzo iniziale sorgente è l'inizio del blocco (1D00); l'indirizzo finale sorgente è invece quello finale del blocco meno la lunghezza del pacchetto di byte che si vogliono ripetere (nel nostro esempio 1 per cui l'indirizzo finale è 1DFE).

Il comando di riempimento del blocco è quindi:

```
*1D01<1D00.1DFEM
```

Il funzionamento di questa procedura è abbastanza semplice. Il primo byte (1D00) viene ricopiato nella seconda posizione (1D01) che costituisce l'indirizzo iniziale destinazione. 1D01 è però anche il secondo byte sorgente che viene allora ricopiato in 1D02; e così avanti. L'ultimo byte ricopiato è 1DFE che va in 1DFF.

Proviamo ora a fare un esempio in cui si vogliono ricopiare i quattro byte: 00, 5E, 7F e FF. Il blocco da riempire è lo stesso dell'esempio precedente. Poniamo per prima cosa i quattro byte nelle prime posizioni:

```
*1D00:00 5E 7F FF
```

Il comando per riempire tutto il blocco è allora:

```
*1D04<1D00.1DFBM
```

Notate che l'indirizzo iniziale di destinazione è il quinto byte del blocco (1D04) e che l'indirizzo finale sorgente è calcolato così (in esadecimale):

```
  1DFF
– 04     (lunghezza del periodo ripetitivo)
------
  1DFB
```

Dopo aver eseguito questi comandi provate a rileggere il blocco di memoria per sincerarvi che essa sia stata riempita nel modo voluto.

Il comando di verifica

Questo comando del Monitor permette di controllare se due blocchi di memoria sono eguali. Nel caso che anche un solo byte sia diverso tra i due blocchi, il Monitor ve lo segnala. Mediante tale comando è possibile quindi verificare se quello che avete scritto su una memoria esterna è eguale a quanto era originariamente in memoria. Infatti dopo avere scritto qualcosa, per esempio su un dischetto, ricopiatelo in memoria in una area diversa. Fate quindi il confronto tra ciò che avevate originariamente in memoria e quanto avete ricopiato; se le due aree sono eguali siete sicuri che il testo sul dischetto è esatto.

Il formato del comando di verifica è analogo a quello del comando di spostamento. Anche in questo caso le due aree di memoria vengono chiamate sorgente e destinazione. Battete allora l'indirizzo iniziale dell'area destinazione, poi il segno di minore (<), poi l'indirizzo iniziale dell'area sorgente, quindi un punto, poi l'indirizzo finale sorgente ed infine una V (per verify).

Ecco un esempio:

```
*32D0<0.CV
```

Il blocco sorgente dall'indirizzo 0 all'indirizzo 000C (gli zeri di sinistra non figurano nel comando), viene confrontato con il blocco destinazione posto tra gli indirizzi 32D0 E 32DC.

Se il Monitor incontra un byte diverso tra blocco sorgente e blocco destinazione, visualizza il suo indirizzo sorgente assieme ai due valori diversi sorgente e destinazione.

Per potere fare un esempio creiamo dapprima due blocchi eguali con il comando di spostamento:

```
*32D0<0.CM
```

Per sicurezza visualizziamo i due blocchi e constatiamo che sono eguali:

```
*0.C

0000- 4C 3C D4 4C 3A DB 8C 8C
0008- FF FF 4C 99 E1
*32D0.32DC
```

```
32D0- 4C 3C D4 4C 3A DB 8C 8C
32D8- FF FF 4C 99 E1
*
```

Cambiamo ora il contenuto del byte 32D8 dal suo valore attuale FF a 5A, con il comando:

```
*32D8:5A
```

Possiamo allora provare ad eseguire il comando di verifica:

```
*32D0<0.CV
```

Il Monitor confronterà byte per byte del blocco sorgente e di quello destinazione e quando troverà diversi contenuti tra l'indirizzo 0008 e 32D8 visualizzerà sullo schermo il seguente messaggio:

```
0008-FF (5A)
*
```

Esso dice appunto che nella posizione di memoria 0008 è contenuto il valore FF, mentre nella posizione di destinazione è contenuto il valore 5A. L'indirizzo di destinazione non viene visualizzato. Per evitare di dovere calcolare l'indirizzo destinazione, potete ripetere il comando di verifica scambiando tra di loro i due blocchi:

```
*0<32D0.32DCV
```

Sullo schermo vi apparirà:

```
32D8-5A (FF)
*
```

Attenzione però che in quest'ultimo caso dovete avere calcolato l'indirizzo finale di destinazione perchè ora ne avete bisogno come indirizzo finale sorgente.

Verifica della scrittura di blocchi di memoria sulle periferiche

Abbiamo già detto che mediante il comando di verifica è possibile controllare quanto viene scritto su una memoria esterna; potete infatti rileggere e poi verificare ciò che avete scritto in precedenza.

Negli esempi che seguono vi mostriamo come effettuare questa procedura per programmi in assembler, per tavole delle forme e per altre cose.

Supponiamo per esempio di volere registrare su cassetta un certo blocco di memoria. Il comando di registrazione è il seguente:

```
*2000.20FFW
```

Il blocco, posto tra gli indirizzi 2000 e 20FF, viene quindi trasferito su una cassetta. Non dimenticatevi però di premere il tasto RECORD del registratore *prima* di dare RETURN! Al termine della registrazione udirete un "beep"; fermate il registratore e riavvolgete la cassetta sino all'inizio della zona del segnale di riferimento. Caricate allora dalla cassetta, quanto avete appena registrato, in una zona di memoria che sapete libera (per esempio tra 2100 e 21FF):

```
*2100.21FFR
```

Anche in questo caso non dimenticatevi di avviare il registratore (PLAY). Al termine della lettura udirete il solito "beep". A questo punto potete dare il comando di verifica:

```
*2000<2100.21FFV
```

Se durante questa operazione il Monitor non vi segnala alcuna differenza tra i due blocchi, potete essere sicuri che quello che avete registrato su cassetta è corretto.

Se invece delle cassette usate i dischetti la procedura è la stessa salvo che ora dovete avere già in memoria il DOS. Il primo comando di registrazione è BSAVE:

```
BSAVE MEMDATA, A$2000,L$FF
```

La lettura del disco viene invece eseguita con il comando BLOAD:

```
BLOAD MEMDATA, A$2100
```

Il file MEMDATA sorgente inizia dall'indirizzo esadecimale 2000, mentre quello destinazione dall'indirizzo, sempre in esadecimale, 2100. Per dare il comando di verifica dovete allora dare le due istruzioni:

```
CALL -151
*2000<2100.21FFV
```

Se il Monitor non vi segnala alcuna differenza, avete allora la sicurezza di avere una registrazione su disco corretta.

IL COMANDO GO

Il Monitor ha un comando che permette di trasferire, il controllo del calcolatore, ad un programma che inizia ad uno specifico indirizzo. All'inizio di questo capitolo avevamo già visto un comando particolare per passare dal Monitor all'Applesoft nella versione disco. Questo comando:

```
*3D0G
```

è appunto un comando "GO" di trasferimento del controllo. Esso istruisce il Monitor di saltare all'indirizzo RD0 e di eseguire l'istruzione che trova in quell'indirizzo. La lettera G significa appunto GO. Nel caso dell'esempio, all'indirizzo 3D0 si trova l'istruzione:

```
JMP $9DB9
```

che è a sua volta una istruzione di salto all'indirizzo 9DB9 dove iniziano le routine del DOS.

Il formato più generale del comando GO richiede l'indirizzo di salto seguito dalla lettera G. L'indirizzo può essere omesso e in tal caso viene usato quello indicato in quel momento dal puntatore di memoria.

USO DELLA STAMPANTE

Se avete una stampante, collegata al calcolatore tramite l'interfaccia seriale o la cartolina per comunicazioni, potete inviarle tutte le informazioni che normalmente vanno sullo schermo. Per commutare l'uscita verso la stampante dovete dare il numero di slot, a cui è collegata la stampante, seguito da CTRL-P e RETURN. In questo modo tutti i messaggi o i dati, che normalmente appaiono sullo schermo, vanno invece sulla stampante.

Per ritornare al normale uso dello schermo, battete il numero 0 del primo slot e poi i comandi CTRL-P e RETURN.

Attenzione però che lo slot che selezionate con questo comando deve contenere una cartolina. Se esso è vuoto, il calcolatore entra in uno stato di attesa da cui non può più uscirne e sarete quindi costretti a premere RESET.

Questo comando di abilitazione della stampante opera esattamente come il comando PR# del BASIC. Ambedue cambiano il contenuto dei due byte dell'interruttore a software CSW (character output switch) posto all'indirizzo 54 ($36). Questi due byte contengono infatti il puntatore alla subroutine, di gestione dei caratteri di uscita, attualmente in uso. Mediante CTRL-P viene cambiata tale subroutine e quindi la periferica di uscita.

IL COMANDO "TASTIERA"

Questo comando permette di ricevere dei dati in ingresso da una periferica diversa dalla tastiera standard del calcolatore. Come nel caso della stampante dovete indicare il numero di slot, a cui è collegata la nuova periferica, e poi battere CTRL-K o RETURN. Per ripristinare il normale uso della tastiera date la stessa sequenza di comandi indicando però lo slot numero 0.

Come nel caso precedente questo comando agisce sull'interruttore a software KSW (keyboard input switch) posto all'indirizzo 56 ($38).

CAMBIAMENTO DEI MODI DI LAVORO DELLO SCHERMO

Per lavorare con lo schermo nel modo inverso, sotto il controllo del Monitor, battete la lettera I. Tutti i dati che il calcolatore visualizzerà sullo schermo appariranno in nero su campo bianco, tranne però le vostre istruzioni al Monitor che rimarranno in modo normale (bianche in campo nero).

Per ritornare al modo normale battete la lettera N.

Ambedue questi comandi richiedono solo le lettere I o N.

USO DELL'ARITMETICA BINARIA A OTTO BIT DEL MONITOR

Il Monitor può effettuare delle operazioni aritmetiche a otto bit. Queste operazioni sono l'addizione e la sottrazione. Attenzione però che anche i risultati sono solo di otto bit per cui se vi è un riporto esso viene troncato.

Per fare l'addizione battete il primo addendo in esadecimale, poi il segno (+) e quindi il secondo addendo anche lui in esadecimale. Se il risultato è superiore a FF il Monitor tronca la cifra più significativa e visualizza gli otto bit di ordine inferiore. Ecco un esempio:

```
*7F+8A
=09
```

Per la sottrazione battete, in modo analogo, il minuendo seguito dal segno (–) e poi il sottraendo. I due valori devono essere esadecimali. Se il risultato è negativo il Monitor visualizza il complemento come in questo esempio:

```
*0A-2D
=DD
```

COMANDO DEL MONITOR DEFINITO DALL'UTENTE

Mediante il comando CTRL-Y è possibile porre in esecuzione un comando definito dall'utente. CTRL-Y fa infatti saltare all'indirizzo esadecimale 3F8 ove è possibile porre una istruzione JMP di salto ad una sobroutine in linguaggio macchina scritta dall'utente.

Mostriamo per esempio come sia possibile passare all'Applesoft, senza usare il solito comando 3D0G, mediante il nuovo comando CTRL-Y.

Per prima cosa dobbiamo usare l'istruzione JMP che richiede tre byte. Il codice di questa istruzione è 4C e viene posto nel primo byte. Negli altri due byte poniamo l'indirizzo destinazione con l'avvertenza che dobbiamo indicare l'ultimo byte dell'indirizzo per primo e poi l'altro.

Facciamo un esempio in cui vogliamo porre 3D0 come indirizzo di destinazione della JMP. Per cui invertendo i due byte possiamo scrivere:

```
*3F8:4C D0 03
```

Provate ora a dare CTRL-Y e vedrete che equivale a 3D0G, anzi è meglio!

Vediamo ora un altro esempio in cui usiamo CTRL-Y per passare al Mini-Assembler che descriveremo in dettaglio nel prossimo paragrafo. Se date il comando:

```
*F666G
```

vi appare il carattere di pronto del Mini-Assembler:

```
!
```

L'indirizzo di partenza del Mini-Assembler è F666 e può essere inserito in una istruzione JMP posta all'indirizzo 3F8:

```
!3F8:JMP $F666

03F8-    4C 66 F6     JMP    $F666 ←— Il Mini-Assembler
!                                      visualizza questa linea
```

Nel prossimo paragrafo vedremo il significato di quest'ultima riga, ma possiamo sin da ora capire che rappresenta in codice proprio la nostra istruzione JMP alla subroutine che inizia alla posizione F666. Per ritornare al Monitor battete il comando:

```
!$FF69G
*
```

Il carattere di pronto del Monitor appare sullo schermo. Se volete ritornare al Mini-Assembler è sufficiente che premiate CTRL-Y.

Per togliere il vostro comando speciale, eseguibile con CTRL-Y, dovete porre una diversa istruzione di salto all'indirizzo 3F8.

IL MINI-ASSEMBLER

Il Mini-Assembler è un linguaggio che vi permette di programmare direttamente in linguaggio macchina. Esso è chiamato Mini perchè si devono indicare gli indirizzi, nei campi operandi delle istruzioni, in maniera esplicita e non in forma simbolica come normalmente si intende per i linguaggi assembler.

Il Mini-Assembler risiede assieme all'Integer BASIC nella stessa ROM. Lo potete usare quindi se avete un calcolatore Apple II standard o un calcolatore Apple II Plus con la cartolina dell'Integer BASIC.

Un'altra caratteristica del Mini-Assembler è quella per cui le istruzioni sono caricate direttamente in linguaggio macchina e non richiedono alcun assemblaggio come per gli assembler simbolici.

In questo capitolo vi descriviamo il Mini-Assembler, ma non vi insegnamo come programmare in assembler. Non descriviamo neanche le istruzioni del microprocessore 6502 che è appunto il microprocessore usato dall'Apple II.

Se quindi non avete dimestichezza con la programmazione in assembler o non conoscete il microprocessore 6502, rimandate la lettura dei paragrafi seguenti.

COME ACCEDERE AL MINI-ASSEMBLER

L'indirizzo iniziale del Mini-Assembler è F666 in esadecimale. Per accedervi dal Monitor date il comando:

```
*F666G
```

Dall'Integer BASIC o dall'Applesoft (nelle versioni a disco o a cassetta) battete invece il comando:

```
CALL -2458
```

La presenza del Mini-Assembler vi viene segnalata da un "beep" e dal punto esclamativo (!) come carattere di pronto.

Errori in ingresso

Il Mini-Assembler vi segnala gli errori che commettete nella battitura delle singole istruzioni. Udite infatti un "beep" e sullo schermo vi appare il carattere (^) sotto il primo carattere errato della istruzione. Inoltre il contatore di posizioni (location counter) non viene incrementato per cui potete ribattere subito l'istruzione.

COMANDI DEL MONITOR IN MINI-ASSEMBLER

Mentre siete sotto il controllo del Mini-Assembler potete sempre eseguire i comandi del Monitor.

È sufficiente che dopo il punto esclamativo (!) del Mini-Assembler battiate il segno di dollaro ($) seguito dal comando del Monitor. Nell'esempio seguente viene dato il comando per la lettura della memoria:

```
!$1CFF

1CFF- E6
!
```

Potete così risparmiare molto tempo per passare dal Mini-Assembler al Monitor e viceversa. Per dare i comandi del Monitor è sufficiente che li facciate precedere dal segno $.

COME LASCIARE IL MINI-ASSEMBLER

Per lasciare il Mini-Assembler, date uno dei comandi del Monitor con il segno di dollaro come prefisso. Per ritornare al Monitor eseguite un salto all'indirizzo FF69 mediante il comando $FF69G.

$CTRL-B o $CTRL-C commutano il calcolatore in BASIC escluso il caso che voi usiate l'Applesoft proveniente da cassetta o da disco. Nel caso dell'Applesoft proveniente da disco usate il comando $3D0G; se l'Applesoft proviene invece da cassetta usate il comando $0G.

FORMATO DELLE ISTRUZIONI

Sebbene non desideriamo entrare nei dettagli della programmazione in assembler, dobbiamo però precisare alcuni aspetti del Mini-Assembler per poterlo usare.

Per prima cosa dobbiamo dire che il Mini-Assembler usa un suo puntatore, per le istruzioni, diverso dal puntatore per la memoria del Monitor. Questo puntatore del Mini-Assembler deve essere inizializzato prima di caricare le istruzioni.

Inoltre bisogna precisare che esistono vari formati per le istruzioni del microprocessore 6502 che essenzialmente dipendono dal tipo di indirizzamento usato.

Il microprocessore 6502 ha infatti undici modi di indirizzamento e solo sei formati per le istruzioni; facciamo ora alcuni esempi di questi formati.

Il primo formato, detto indirizzamento assoluto o diretto, richiede solo il byte, o i due byte, dell'indirizzo di memoria dell'operando. Per esempio:

```
AND $303A
```

Ricordiamo che il Mini-Assembler non richiede il segno di dollaro ($) prima degli indirizzi esadecimali, in quanto presume che essi siano sempre esadecimali.

Il secondo formato riguarda il modo di indirizzamento immediato, come mostriamo in questo esempio:

```
LDA #$04
```

Il segno di cancelletto (#) è il primo carattere dell'operando. Esso indica che il valore 04 deve essere caricato lui stesso, cioè in maniera immediata, nell'accumulatore. Senza il segno di cancelletto, il Mini-Assembler avrebbe interpretato l'istruzione nel senso di prendere il contenuto della posizione 04 e poi di caricarlo nell'accumulatore.

A questo punto è importante precisare che la parola *immediato,* che usiamo in questo caso, non ha nulla a che fare con il modo di esecuzione immediata del BASIC. Questo attributo di immediato viene usato in due contesti così diversi, il BASIC e i formati delle istruzioni, per cui non vi è alcun pericolo di fare della confusione.

Il terzo formato di indirizzamento è detto a indice e si presenta in questo modo:

```
CMP $23,X
```

oppure:

```
AND $80,Y
```

In ambedue queste istruzioni i registri X e Y appaiono come secondi operandi. Quando il calcolatore eseguirà queste istruzioni, calcolerà gli operandi effettivi come somma del contenuto dei registri, X o Y, più il valore indicato del primo operando.

Il successivo formato viene chiamato indiretto pre-indice.

Ecco un esempio:

```
AND ($F0,X)
```

Esso indica che la somma dell'indirizzo ($F0) e del registro (X) punta ad una posizione di memoria, contenuta nelle prime 256, che a sua volta punta all'operando effettivo dell'istruzione.

Il formato indiretto post-indice ha invece la forma seguente:

```
ORA ($22),Y
```

Il primo operando ($22) rappresenta una posizione di memoria il cui contenuto viene sommato al contenuto del registro Y; il risultato rappresenta l'indirizzo effettivo. Fate attenzione che il Mini-Assembler riconosce e distingue questi due ultimi formati dalla posizione delle parentesi.

L'ultimo formato è l'indirizzamento indiretto. Ecco un esempio:

```
JMP ($22FE)
```

L'operando effettivo è costituito dai due byte posti all'indirizzo $22FE. In altre parole il valore posto tra parentesi è da vedersi come un puntatore piuttosto che come un operando diretto.

USO DEL MINI-ASSEMBLER

Come abbiamo già detto il Mini-Assembler ha un suo puntatore, chiamato contatore delle posizioni, che viene incrementato della lunghezza in byte di ogni nuova istruzione che caricate. In altre parole, ogni volta che una istruzione entra nel calcolatore e viene assemblata in linguaggio macchina, il Mini-Assembler ne calcola la lunghezza in byte (1, 2 o 3 byte) e incrementa di tale valore il contatore di posizioni.

La prima cosa che dovete fare, lavorando con il Mini-Assembler, è definire il valore iniziale nel contatore delle posizioni. Per esempio:

```
!8DB0:LDA #$04
```

pone nel contatore il valore 8DB0. Subito dopo il carattere di pronto del Mini-Assembler (!) battete quindi il valore dell'indirizzo iniziale del vostro programma (nell'esempio 8DB0) seguito da due punti (:) e dalla prima istruzione in assembler. Successivamente non sarà più necessario che battiate gli altri valori del contatore delle posizioni perchè esso si incrementa automaticamente. Per ripristinare o azzerare il contatore (cioè farne il reset) imponetegli un nuovo valore come avete appena visto fare per impostare il primo indirizzo.

Caricate quindi le altre istruzioni una per linea. Dopo la prima, le altre istruzioni devono essere precedute da uno spazio vuoto che serve appunto per indicare al Mini-Assembler di calcolare il nuovo valore del contatore delle posizioni:

```
! JSR FB1E
```

Un esempio di programma

Facciamo ora un esempio di uso del Mini-Assembler in cui descriviamo punto per punto tutte le operazioni che si devono compiere.

In questo esempio vengono usati i due controllori per giochi (i paddle) che danno dei valori d'ingresso per regolare il funzionamento dell'altoparlante. Nel programma si fa uso della subroutine PREAD, contenuta nel Monitor all'indirizzo FB1E. Il controllore 0 fornisce l'intervallo di tempo tra due "click" dell'altoparlante (0 per l'intervallo più corto e FF per quello più lungo). Il controllore 1 fornisce invece un valore inverso rispetto a quello dell'altro controllore (0 per l'intervallo più lungo e FF per quello più corto).

Il programma inizia all'indirizzo 1D00 e usa l'indirizzo 1CFF per memorizzare il riferimento dato dal controllore 0.

Per ogni istruzione in assembler che date, il calcolatore vi risponde con il valore attuale del contatore delle posizioni, con il codice operativo e gli operandi in linguaggio macchina (questa parte viene anche chiamata *codice oggetto*) e poi viene ripetuta l'istruzione stessa in forma simbolica. Ecco un esempio:

```
1D00-    A2 00          LDX    #$00
```

Come vedete il valore iniziale del contatore delle posizioni (1D00) è seguito da un trattino (-) poi viene il codice A2, che si riferisce all'istruzione LDX, seguito dall'operando 00. Nel caso che l'operando richieda due byte, il byte di ordine basso appare prima di quello di ordine alto.

Ecco il programma che vi presentiamo come esempio:

```
!1D00:LDX #$00          ← Date il valore del contatore delle posizioni
                                          e la prima istruzione
1D00-    A2 00          LDX    #$00
! JSR FB1E              ← Tutti i numeri sono in esadecimale (il prefisso $
                                          non è necessario)
```

```
1D02-    20 1E FB     JSR    $FB1E
! STY 1CFF

1D05-    8C FF 1C     STY    $1CFF
! INX

1D08-    E8           INX
! JSR FB1E

1D09-    20 1E FB     JSR    $FB1E
! LDA C030

1D0C-    AD 30 C0     LDA    $C030
! DEC 1CFF

1D0F-    CE FF 1C     DEC    $1CFF
! BNE 1D0C ← Il Mini-Assembler calcola il salto relativo (F8)

1D12-    D0 F8        BNE    $1D0C
! LDA C030

1D14-    AD 30 C0     LDA    $C030
! INY

1D17-    C8           INY
! BNE 1D14

1D18-    D0 FA        BNE    $1D14
! JMP 1D00

1D1A-    4C 00 1D     JMP    $1D00
!
```

Dopo che avete caricato questo programma dovete controllare se avete commesso errori. Fatene quindi un listato, cioè rileggetelo dalla memoria, usando il Monitor come vi abbiamo insegnato in precedenza.

Se infine lo ritenete opportuno, potete registrare questo programma in assembler su una cassetta, mediante il comando W del Monitor, oppure su un disco con il comando BSAVE del BASIC.

Per porre in esecuzione il programma eseguite un salto alla posizione 1D00, mediante il comando G del Monitor oppure con l'istruzione CALL 7424 del BASIC. Provate poi a ruotare le manopole dei due controllori paddle per vedere come si comporta il programma.

Per terminare il programma premete RESET.

LISTATO DISASSEMBLATO

Il Monitor contiene il comando L che permette di listare un programma in linguag-

gio macchina anche se il vostro calcolatore Apple II non ha il Mini-Assembler nella memoria ROM. Il comando L *disassembla* 20 istruzioni in linguaggio macchina in istruzioni in assembler e poi le visualizza. Il comando L usa il contatore di posizioni come puntatore alla istruzione successiva da disassemblare. Di conseguenza se date il comando L, subito dopo aver caricato il programma precedente, l'operazione di disassembler inizierà dall'indirizzo 1D1D e non potrete quindi avere il listato del vostro programma.

È necessario quindi che posizionate il contatore di posizioni all'inizio del programma. Ecco come fare nel nostro caso:

```
!$1D00L
1D00-   A2 00       LDX    #$00
1D02-   20 1E FB    JSR    $FB1E
1D05-   8C FF 1C    STY    $1CFF
1D08-   E8          INX
1D09-   20 1E FB    JSR    $FB1E
1D0C-   AD 30 C0    LDA    $C030
1D0F-   CE FF 1C    DEC    $1CFF
1D12-   D0 F8       BNE    $1D0C
1D14-   AD 30 C0    LDA    $C030
1D17-   C8          INY
1D18-   D0 FA       BNE    $1D14
1D1A-   4C 00 1D    JMP    $1D00
1D1D-   9F          ???
1D1E-   4E A5 12    LSR    $12A5
1D21-   A4 96       LDY    $96
1D23-   A3          ???
1D24-   D0 A4       BNE    $1CCA
1D26-   EF          ???
1D27-   A4 62       LDY    $62
1D29-   A2 70       LDX    #$70
```

Notate che le ultime otto istruzioni disassemblate sono prive di significato in quanto il nostro programma termina all'indirizzo 1D1A.

Vi ricordiamo ancora che il comando L fa parte del Monitor, per cui dovete farlo precedere dal segno di dollaro $ se lavorate con il Mini-Assembler. Se quindi battete L (o $L con il Mini-Assembler) e poi RETURN, senza cambiare il valore del contatore di posizioni, il Monitor vi disassembla le 20 istruzioni successive. Potete allora disassemblare un programma lungo quanto volete analizzando 20 istruzioni alla volta.

PROVA E CORREZIONE DEI PROGRAMMI

Il Monitor di un calcolatore Apple II standard vi permette, oltre ad usare il Mini-Assembler, di potere effettuare la correzione di un programma in assembler. Quando lavorate con un linguaggio a basso livello, come l'assembler, è molto difficile farne la correzione in quanto bisogna controllare sia la struttura logica del programma, come anche il contenuto di molte posizioni di memoria e dei registri.

I comandi del Monitor che vi permettono di fare la correzione di un programma in assembler sono Step e Trace.

I comandi Step e Trace non sono disponibili nei calcolatori Apple II dotati di Autostart Monitor.

Il comando Step

Per controllare un programma in assembler è spesso conveniente provare ad eseguirlo istruzione per istruzione, purchè il programma stesso non sia troppo lungo. Il comando Step permette di eseguire una istruzione alla volta e di controllare il contenuto dei registri.

Con il comando Step il Monitor esegue quindi le seguenti operazioni. Dapprima disassembla e visualizza l'istruzione puntata dal contatore di posizioni, poi la esegue e visualizza il contenuto dei registri del microprocessore. Infine passa il controllo al Monitor per permettervi di dare un nuovo comando.

Il formato del comando Step è una lettera S preceduta da un eventuale indirizzo con cui inizializzare il contatore di posizioni.

Proviamo per esempio ad applicare il comando Step al programma musicale che abbiamo visto poco sopra e che inizia dall'indirizzo 1D00:

```
*1D00S

1D00-    A2 00         LDX    #$00
 A=FF X=00 Y=8C P=32 S=F8
*
```

Dopo un comando Step potete dare un altro comando eguale oppure un comando diverso come per esempio quello di lettura della memoria. Proviamo per esempio ad arrivare all'istruzione STY del nostro programma (indirizzo 1D05). Per arrivare ad essa dovete dare 9 volte il comando S in quanto l'istruzione JSR vi fa passare all'indirizzo FB1E prima di ritornare alla posizione 1D05. Quando arrivate a questa posizione provate allora a leggere la posizione di memoria 1CFF in cui dovreste trovare la lettura del paddle 0.

```
*S

1D05-    8C FF 1C      STY    $1CFF
 A=00 X=00 Y=00 P=32 S=F8
*1CFF

1CFF- 00
*
```

Il comando Trace

Il comando Trace è analogo a Step in quanto visualizza gli stessi dati, ma non richiede che venga battuto un comando per ogni istruzione. Il comando Trace opera automaticamente finchè non viene fermato.

Per terminare la visualizzazione del comando Trace vi sono due possibilità. La prima è quella di premere RESET; la seconda è quella di inserire nel programma stesso una istruzione BRK. Questa istruzione BRK, quando verrà eseguita, passerà il controllo alla tastiera fermando quindi l'esecuzione del programma.

Il formato del comando Trace è costituito da una lettera T preceduta eventualmente dal valore da porre nel contatore delle posizioni. Ecco un esempio che si riferisce al programma descritto precedentemente:

```
*1D00T

1D00-   A2 00       LDX    #$00
 A=FF X=00 Y=8C P=32 S=F6
1D02-   20 1E FB    JSR    $FB1E
 A=FF X=00 Y=8C P=32 S=F6
FB1E-   AD 70 C0    LDA    $C070
 A=00 X=00 Y=8C P=32 S=F4
FB21-   A0 00       LDY    #$00
 A=00 X=00 Y=00 P=32 S=F4
FB23-   EA          NOP
 A=00 X=00 Y=00 P=32 S=F4
FB24-   EA          NOP
 A=00 X=00 Y=00 P=32 S=F4
FB25-   BD 64 C0    LDA    $C064,X
 A=00 X=00 Y=00 P=32 S=F4
FB28-   10 04       BPL    $FB2E
 A=00 X=00 Y=00 P=32 S=F4
FB2E-   60          RTS
 A=00 X=00 Y=00 P=32 S=F4
1D05-   8C FF 1C    STY    $1CFF
 A=00 X=00 Y=00 P=32 S=F6
1D08-   E8          INX
 A=00 X=01 Y=00 P=30 S=F6
1D09-   20 1E FB    JSR    $FB1E
 A=00 X=01 Y=00 P=30 S=F6
FB1E-   AD 70 C0    LDA    $C070
 A=27 X=01 Y=00 P=30 S=F4
FB21-   A0 00       LDY    #$00
 A=27 X=01 Y=00 P=32 S=F4
FB23-   EA          NOP
 A=27 X=01 Y=00 P=32 S=F4
FB24-   EA          NOP
 A=27 X=01 Y=00 P=32 S=F4
FB25-   BD 64 C0    LDA    $C064,X
 A=27 X=01 Y=00 P=30 S=F4
FB28-   10 04       BPL    $FB2E
 A=27 X=01 Y=00 P=30 S=F4
FB2E-   60          RTS
```

Il comando Trace presenta però lo svantaggio, rispetto a Step, di offrire meno possibilità di intervento dopo l'esecuzione delle singole istruzioni del programma. Inoltre dovete prevedere di porre nel programma, sin dall'inizio, le istruzioni BRK che dovranno poi essere sostitute da istruzioni fittizie NOP (No Operation) quando avrete terminato la correzione del programma.

Un altro aspetto riguarda la velocità di esecuzione di un programma sotto il controllo del comando Trace che diviene molto più lenta. Provate per esempio a porre una istruzione BRK nella posizione 1D1A e poi ad eseguire il programma senza Trace e con il Trace. Nel primo caso date il comando 1D00G e nel secondo invece il comando 1D00T. Vi accorgerete subito che in condizioni normali questa piccola porzione di programma viene eseguita in una frazione di secondo, mentre con il Trace richiede 60 o 70 secondi. Il consiglio, che possiamo darvi quindi, è quello di usare con molta cautela il comando Trace per evitare che programmi molto lunghi richiedano tempi eccessivi di esecuzione.

Alcune avvertenze sul contatore di posizioni

Come abbiamo detto in precedenza è possibile dare la maggior parte dei comandi del Monitor assieme a Step e Trace. Ci sono però alcune eccezioni che riguardano i comandi List, Go e CTRL-Y (definito dall'utente) in quanto questi comandi modificano il contatore delle posizioni. Di conseguenza se date uno di questi comandi interrompete il normale fluire del programma salvo che non ripristiniate il corretto valore del contatore delle posizioni dopo l'esecuzione di uno di essi. Qui di seguito vi diamo un esempio in cui si vede come dopo il comando List venga alterato il contatore delle posizioni:

```
*1D00S
 1D00-    A2 00        LDX    #$00
  A=62 X=00 Y=00 P=32 S=F8
 *S
 1D02-    20 1E FB     JSR    $FB1E
  A=62 X=00 Y=00 P=32 S=F8
 *L                                        ← Comando di LIST

FB1E-    AD 70 C0     LDA    $C070
FB21-    A0 00        LDY    #$00
FB23-    EA           NOP
FB24-    EA           NOP
FB25-    BD 64 C0     LDA    $C064,X
FB28-    10 04        BPL    $FB2E
FB2A-    C8           INY
FB2B-    D0 F8        BNE    $FB25
FB2D-    88           DEY
FB2E-    60           RTS
FB2F-    A9 00        LDA    #$00
FB31-    85 48        STA    $48
```

```
FB33-   AD 56 C0    LDA   $C056
FB36-   AD 54 C0    LDA   $C054
FB39-   AD 51 C0    LDA   $C051
FB3C-   A9 00       LDA   #$00
FB3E-   F0 0B       BEQ   $FB4B
FB40-   AD 50 C0    LDA   $C050
FB43-   AD 53 C0    LDA   $C053
FB46-   20 36 F8    JSR   $F836
*S                                ← Cambiate il contatore
                                    delle posizioni
FB49-    A9 14        LDA   #$14    così che al prossimo passo
 A=14 X=00 Y=00 P=30 S=F6           sarà a $FB49 invece di $FB1E
*S

FB4B-    85 22        STA   $22
 A=14 X=00 Y=00 P=30 S=F6
*
```

Subroutine del Monitor

Spesso il BASIC non è così potente come si vorrebbe per potere realizzare tutte le funzioni che si desiderano. Per questo motivo vengono spesso scritte in assembler delle opportune subroutine per arricchire le possibilità del BASIC.

Attenzione però che non sempre è così facile, come si potrebbe pensare, usare delle subroutine in assembler. Per esempio dobbiamo stabilire dove caricarle in memoria e come passare loro i parametri necessari. Ricordiamo poi che nella memoria dell'Apple II vi sono quattro grandi aree riservate per le due pagine grafiche in bassa risoluzione, o per i testi, e per le due pagine grafiche in alta risoluzione. Anche il DOS e l'interprete Applesoft occupano della memoria. Si può in generale dire che le zone libere della memoria, dove caricare l'assembler, dipendono dalla sua ampiezza e dal tipo di calcolatore che avete.

Normalmente il Monitor è la maggior fonte di subroutine in assembler per tre motivi principali. Primo perchè sono contenute nella ROM e non dovete quindi preoccuparvi della loro rilocazione; secondo perchè sono state già corrette e controllate e terzo perchè non richiedono altra memoria RAM. Queste subroutine sono riportate nell'Appendice D.

Utilizzo delle subroutine

Se decidete di usare le subroutine del Monitor, controllate prima di tutto che non vi siano delle funzioni equivalenti nel BASIC per evitare di fare le cose più complicate del necessario. Secondo controllate se dovete trasmettere dei parametri alle subroutine o se ne dovete ricevere alla loro fine. Questo passaggio di parametri richiede normalmente che vengano posti dei valori nei registri del microprocessore prima di eseguire la subroutine oppure che i registri vengano letti al termine della sua esecuzione. Per fare questo è necessario allora che vengano usate altre istruzioni in assembler per interfacciare il BASIC con la subroutine. Attenzione però che la maggior

parte delle subroutine non richiede parametri dal BASIC; quelle che li dovessero richiedere hanno già una equivalente funzione nel BASIC stesso.

Quando inserite poi le chiamate delle subroutine nel vostro programma, lo dovete fare in maniera chiara e documentata. Per esempio CALL -936 pulisce lo schermo e porta il cursore in alto a sinistra. Un modo per rendere le cose più chiare è quello di assegnare ad una variabile l'indirizzo di chiamata:

```
10 CLSCREEN=-936
```

e poi usare questa variabile nel resto del programma:

```
1510 CALL CLSCREEN
```

Questo rende sicuramente chiaro il contesto a chi dovesse leggere il vostro programma a voi stessi se lo doveste correggere.

Problemi da evitare

Se avete a vostra disposizione un editor/assemblatore per l'Apple II, sarebbe molto facile rilocare le subroutine per poterle usare in calcolatori Apple II con differenti ampiezze di memorie RAM. In caso contrario dovete voi stessi riscrivere le subroutine e potreste così occupare degli indirizzi riservati per il DOS, o per una pagina grafica, o per l'Applesoft. Per questo motivo cercate di usare solo le subroutine del Monitor.

Se lavorate in Applesoft, usate la funzione USR per passare i parametri ad una subroutine, e non l'istruzione CALL. Gli indirizzi da 9D a A3 memorizzano i valori passati da USR e voi potete usare questa area per i parametri da passare indietro al BASIC. Usate poi l'istruzione POKE per porre un'istruzione JMP negli indirizzi da 10 a 12 (da 0A a 0C in esadecimale). Questi indirizzi devono infatti contenere una istruzione JMP di salto all'inizio della subroutine in assembler richiamata da USR.

INSERIMENTO DEI VOSTRI PROGRAMMI NEL BASIC

Le istruzioni LOMEM: e HIMEM: proteggono il vostro programma in assembler da collisioni con il BASIC. Dovete però prestare una certa attenzione se volete lavorare in assembler e contemporaneamente avere l'Applesoft, proveniente da disco o da cassetta, e il DOS. In generale per lavorare contemporaneamente con le subroutine in assembler, con il BASIC e con il DOS dovete procedere così:

1. Caricare il DOS.
2. Caricare, se necessario, l'Applesoft da disco o da cassetta.
3. Porre i valori di LOMEM: e HIMEM:.
4. Caricare la subroutine in assembler.
5. Caricare il programma in BASIC.

Il DOS cambia il valore di HIMEM: mentre viene caricato in memoria. L'Applesoft sposta invece il valore di LOMEM:. Voi definite quindi i valori di LOMEM: e HIMEM: per fare posto alla subroutine in assembler. Successivamente, quando caricherete ed eseguirete il programma in BASIC, userete solo la memoria che è rimasta disponibile tra LOMEM: e HIMEM:.

Per ulteriori chiarimenti, su dove potete inserire le subroutine in assembler, vi rimandiamo alla mappa di memoria dell'Appendice G. Consultate anche le voci HIMEM: e LOMEM: del capitolo 8.

CAPITOLO 8

RIEPILOGO DELLE ISTRUZIONI E DELLE FUNZIONI BASIC

In questo capitolo viene descritta la sintassi di tutte le istruzioni e di tutte le funzioni del BASIC dell'APPLE II. Sia le istruzioni che le funzioni sono descritte in ordine alfabetico.

Nei capitoli dal 3 al 7 vi abbiamo presentato i concetti fondamentali della programmazione facendo uso sia di istruzioni che di funzioni; ora vi descriviamo invece, dettagliatamente, le regole per l'uso delle singole istruzioni e funzioni.

MODO IMMEDIATO E MODO DIFFERITO

La maggior parte delle istruzioni possono essere usate sia in modo immediato che differito. Se non vi viene data alcuna indicazione contraria, potete sempre ritenere di usarle in ambedue i casi. Nei casi particolari, in cui una istruzione può essere usata solo in uno dei due modi, vi verrà espressamente segnalato.

Alcune istruzioni fanno parte del sistema operativo a disco DOS e possono essere usate in modo immediato. Potete usarle anche in modo differito, ma dovete inserirle in una PRINT sotto forma di stringa precedute da CTRL-D (codice ASCII 4). Queste due istruzioni sono infatti equivalenti:

```
]CATALOG
]PRINT CHR$(4); "CATALOG"
```

Notate che invece di usare CHR$ (4) avreste potuto porre tra virgolette il carattere CTRL-D. In tal caso sullo schermo avreste avuto solamente le virgolette perchè CTRL-D non appare con alcun simbolo.

VERSIONI DEL BASIC

Se non viene dato un avviso contrario tutte le istruzioni e le funzioni operano egualmente sia con l'Integer BASIC che con l'Applesoft. Eventuali differenze vi verranno di volta in volta segnalate.

CRITERI PER INTERPRETARE I FORMATI

Qui di seguito riportiamo i criteri con cui saranno descritti i formati delle istruzioni e delle funzioni.

{ }	Le parentesi graffe indicano la possibilità di scelta tra diversi elementi. Almeno uno degli elementi deve essere scelto. Le parentesi non compariranno nell'effettiva istruzione.
[ ]	Le parentesi quadre indicano un parametro opzionale. Le parentesi non compariranno nell'effettiva istruzione.
	I puntini indicano che l'elemento precedente può essere ripetuto. I puntini stessi non compariranno nell'effettiva istruzione.
numero di linea	Tutte le istruzioni in modo differito devono iniziare con un numero di linea.
altri segni di interpunzione	Tutti gli altri segni di interpunzione come: virgola, punto e virgola, virgolette e parentesi devono essere riportati così come sono indicati.
LETTERE MAIUSCOLE	Lettere e parole in maiuscolo devono essere riportate così come sono indicate.
caratteri italici	I termini generici sono in *italico.* Il programmatore dovrà fornire il termine esatto in accordo con le definizioni riportate qui di seguito.

I termini qui di seguito riportati, con i caratteri italici, sono comuni a tutte le istruzioni e le funzioni. Eventuali altri termini in italico possono comparire solo in alcune istruzioni; in tal caso verranno definiti di volta in volta.

col	Indica il numero di una colonna in un grafico a bassa risoluzione; può essere compreso tra 0 e 39.
colh	Indica il numero di una colonna in un grafico ad alta risoluzione; può essere compreso tra 0 e 279.
cost	Indica una costante numerica oppure di stringa.
D*n*	È il numero di un drive di unità a disco; può essere D0 o D1.
espr	Una costante, o variabile, o espressione sia di tipo numerico che di stringa, anche relazionale o Booleana (solo in Applesoft).
espr$	Una costante, o variabile, o espressione solo di tipo stringa.

esprnm	Una costante, o variabile, o espressione solo di tipo numerico.
nomefile	Indica il nome di un file.
linea	Indica il numero di linea di una istruzione BASIC.
linea$_i$	Indica la linea i-esima di un programma BASIC.
memadr	Una espressione o una variabile o una costante numerica che individua un indirizzo di memoria. L'indirizzo può essere compreso tra –65535 e 65535 (valori decimali); dove –65535 corrisponde all'indirizzo 1, –65534 all'indirizzo 2, ecc.
memloc	Un indirizzo di memoria definito da una costante intera compresa tra 0 e 65535 (valori decimali) o $O e $FFFF (valori esadecimali). I valori esadecimali sono caratterizzati dal segno di dollaro ($) come prefisso.
messaggio	Un testo di stringa racchiuso tra virgolette.
rig	Indica il numero di una riga in un grafico a bassa risoluzione; può essere compreso tra 0 e 47.
righ	Indica il numero di una riga in un grafico ad alta risoluzione; può essere compreso tra 0 e 191.
S*n*	Numero di slot; può essere S0, S1, S2, S3 S4, S5, S6 o S7.
var	In Integer BASIC rappresenta una variabile di stringa o numerica. In Applesoft rappresenta una variabile di stringa o numerica o intera.
varnm	Rappresenta il nome di una variabile numerica.
var (ind)	In Integer BASIC rappresenta una variabile numerica con indici. In Applesoft rappresenta una variabile con indici numerica, intera o di stringa.
V*n*	Rappresenta un numero di volume su disco compreso tra V0 e V255.

ISTRUZIONI

APPEND

Apre un file (vedere OPEN) e posiziona il puntatore del file alla sua fine.

Formato:

APPEND *nomefile* [, D*n*] [, S*n*] [, V*n*]

Tabella 8.1 — Subroutine in linguaggio macchina per APPEND

Linguaggio macchina		Assembler 6502	
Decimale	**Esadecimale**	**Istruzione**	**Commento**
169 0 76 237 253	A9 0 4C ED FD	LDA $0 JMP $FDED	La routine del Monitor $FDED fa uscire, il carattere nel registro A ($0 in questo caso), verso il dispositivo di uscita selezionato, il disco. Vedere l'Appendice D.

Un buffer di memoria di 595 byte viene riservato per il file richiamato. Il file deve essere il tipo sequenziale. Mediante il comando WRITE viene scritta della nuova informazione nel file subito dopo l'ultimo carattere scritto (salvo che non vi siano byte non utilizzati nel mezzo).

Attenzione che il comando APPEND può iniziare ad aggiungere nuovi dati non in coda al file, ma al suo inizio! Per evitare questo gravissimo inconveniente dovete sempre scrivere un carattere di fine file prima di chiuderlo in fase di scrittura. In Tabella 8.1 riportiamo una piccola routine per scrivere il carattere di fine file. Cercate di tenere in memoria questa routine in un punto in cui siano disponibili cinque byte liberi (per esempio nelle posizioni da 768 a 772). La routine potrà essere poi richiamata con una istruzione CALL prima di chiudere il file.

Se il file non esiste nel drive D*n* dello slot S*n*, apparirà il messaggio FILE NOT FOUND. Se il disco nel drive D*n* dello slot S*n* non è il volume V*n* causate l'errore VOLUME MISMATCH. V0 va in accordo con qualunque disco. Se il file è già aperto, APPEND lo chiude per poi riaprirlo (vedere CLOSE).

D*n*, S*n* e V*n* possono essere dati con qualunque ordine.

Se D*n* e S*n* sono omessi viene imposto l'ultimo valore di drive e di slot usati. Se V*n* è omesso, viene imposto V0. Anche il valore *n* può essere omesso e sostituito con D0, S0 e V0.

Siccome APPEND è un comando DOS, richiede un carattere CTRL-D in una PRINT data in modo differito.

APPEND non può essere usato in modo immediato.

AUTO

Permette la numerazione automatica delle linee di programma in Integer BASIC.

Formato:

AUTO *linea* [, *incremento*]

I numeri di linea vengono visualizzati automaticamente, appena premuto RETURN, dal valore iniziale *linea* per *incrementi* successivi. Se l'incremento non è indicato viene imposto il valore 10. CTRL-X cancella un numero visualizzato automaticamente. Dopo CTRL-X si può dare MAN per annullare il comando AUTO (vedere MAN).

Può essere usato solo in modo immediato.

Non è disponibile in Applesoft.

BLOAD

Carica un file binario dal disco nella memoria a partire da un certo indirizzo.

Formato:

BLOAD *nomefile* [, A *memloc*] [, D*n*] [, S*n*] [, V*n*]

Se il parametro A è assente, il file viene caricato a partire dall'indirizzo da cui fu trasferito sul disco (vedere BSAVE). Se A è presente il file viene caricato dall'indirizzo *memloc.*

Fare molta attenzione perché il file viene sovrapposto in memoria sopra qualunque altra cosa sia già presente (per esempio l'interprete Applesoft o il DOS).

Se il file non esiste nel drive D*n* dello slot S*n*, si genera l'errore FILE NOT FOUND. Se non vi è corrispondenza del parametro di volume V*n*, si genera l'errore VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* e S*n* sono omessi vengono imposti gli ultimi valori di drive e slot usati. Se V*n* è omesso viene imposto V0.

BLOAD è un comando DOS e richiede il carattere CTRL-D in una PRINT data in modo differito.

BRUN

Carica un file binario (che deve essere un programma in linguaggio macchina) e lo pone in esecuzione mediante una istruzione JMP dell'assembler del microprocessore 6502.

Formato:

BRUN *nomefile* [, A *memloc*] [, D*n*] [, S*n*] [, V*n*]

Se il parametro A è assente il file viene caricato in memoria dall'indirizzo da cui fu trasferito su disco (vedere BSAVE). Se A è presente, il file viene caricato dall'indirizzo *memloc.*

Un programma in linguaggio macchina richiede normalmente di essere caricato a partire da uno specifico indirizzo. Dovete quindi sempre controllare se un programma

ha istruzioni dipendenti da indirizzi assoluti, prima di caricarlo in una nuova zona di memoria. Il comando BRUN sovrappone il file su ciò che era già in memoria per cui rischiate di distruggere il software operativo.

Se il file non esiste nel drive D*n* nello slot S*n*, vi viene dato l'errore FILE NOT FOUND. Se non vi è corrispondenza del numero di volume V*n* vi viene segnalato l'errore VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* e S*n* sono omessi vengono imposti gli ultimi valori di drive e slot usati. Se V*n* è omesso viene imposto V0.

BRUN è un comando DOS per cui richiede CTRL-D in una PRINT data in modo differito.

BSAVE

Salva un file su disco, cioè trasferisce una parte della memoria su disco.

Formato:

BSAVE *nomefile*, A *memloc*, L *lungh* [, D*n*] [, S*n*] [, V*n*]

Il parametro A indica l'indirizzo di memoria iniziale della zona da salvare. Il parametro L specifica il numero di byte da salvare; *lungh* deve essere un intero compreso tra 0 e 32767 (decimale). Può essere dato anche in forma esadecimale anteponendo il segno di $.

Se non vi è corrispondenza del numero di volume V*n* viene segnalato l'errore VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* e S*n* sono omessi, vengono imposti gli ultimi valori di drive e di slot usati. V0 viene imposto se V*n* è assente. Se *n* è assente vengono imposti i valori D0, S0 e V0.

BSAVE è un comando DOS per cui richiede CTRL-D in una PRINT data in modo differito.

CALL

Richiama una subroutine in linguaggio macchina mediante un salto ad un certo indirizzo.

Formato:

CALL *memadr*

CALL può essere usata sia con le subroutine scritte da voi, sia con quelle fornite con l'Apple II (vedere Appendice D).

CATALOG

Visualizza la lista di tutti i nomi di file presenti su un disco.

Formato:

CATALOG [, D*n*] [, S*n*]

CATALOG visualizza DISK VOLUME seguito dal numero di volume. Se il parametro V*n* fosse inserito nel comando, verrebbe ignorato.

Subito dopo viene visualizzata la lista dei nomi dei file sul disco. Per ogni nome viene indicato il tipo di file e il numero di settori occupati. Se un file è protetto (vedere LOCK) appare alla sua sinistra un asterisco. I tipi di file sono i seguenti:

I	Programma in Integer BASIC
A	Programma in Applesoft
T	Testo
B	File binario in linguaggio macchina

Se la lunghezza di un file supera 255 settori, viene indicato il valore modulo 255. Cioè invece di 256 appare il valore 0, invece di 257 il valore 1, ecc.

D*n* e S*n* possono essere dati con qualunque ordine. Se D*n* e S*n* sono omessi viene imposto l'ultimo valore usato.

CATALOG è un comando DOS per cui richiede CTRL-D in una PRINT data in modo differito.

CHAIN

Carica ed esegue un programma in Integer BASIC da un disco, senza annullare i valori delle vecchie variabili.

Formato:

CHAIN *nomefile* [, D*n*] [, S*n*] [, V*n*]

Questo comando può essere usato, solo in Integer BASIC, per caricare un programma in Integer BASIC.

Se il file non esiste nel drive D*n* dello slot S*n*, viene visualizzato il messaggio FILE NOT FOUND. Se non vi è corrispondenza del numero di volume V*n* viene segnalato l'errore VOLUME MISMATCH.

D*n*, S*n*, e V*n* possono essere dati con qualunque ordine. Se D*n* o S*n* sono omessi, vengono imposti i valori usati per ultimo. V0 viene usato in caso dl V*n* mancante. Se *n* non è specificato vengono imposti i valori D0, S0 e V0.

CHAIN è un comando DOS per cui richiede CTRL-D in una PRINT data in modo differito.

CLEAR

Questa istruzione Applesoft assegna il valore 0 a tutte le variabili numeriche e il valore nullo a quelle di stringa.

Formato:

CLEAR

Eseguire CLEAR equivale a spegnere, e riaccendere il calcolatore, e poi a ricaricare il programma. Un programma può continuare a girare dopo CLEAR purchè non ne venga modificata la logica di funzionamento.
In Integer BASIC usate invece CLR.

CLR

Questa istruzione dell'Integer BASIC assegna il valore zero alle variabili numeriche e quello nullo alle variabili di stringa.

Formato:

CLR

Toglie anche il dimensionamento a tutte le variabili con indici e alle stringhe. Potete ancora visualizzare i valori di queste variabili con indici dopo CLR, purché non siano stati assegnati dei valori alle variabili nel frattempo.
CLR può essere usato solo in modo immediato.
In Applesoft usate CLEAR.

CLOSE

Toglie il buffer, usato per un file su disco, e se l'ultima operazione era WRITE, salva su disco il contenuto di questo ultimo buffer.

Formato:

CLOSE [*nomefile*]

Dovete sempre chiudere un file, dopo operazioni di scrittura, per evitare di perdere dei dati. Se il nome del file è presente viene chiuso solo quel file; diversamente vengono chiusi tutti i file (eccetto i file EXEC).
Se un file sequenziale riempie esattamente il buffer al momento della chiusura, allora una successiva istruzione APPEND aggiungerà i nuovi dati all'inizio del file e non alla sua fine. Per evitare ciò, eseguite la routine della Tabella 8.1 prima di chiudere il file. Tale routine può essere tenuta in memoria ove vi siano cinque byte liberi (come tra 768 e 772).

CLOSE è un comando DOS per cui richiede CTRL-D in una PRINT data in modo differito.

COLOR =

Sceglie un colore per i grafici a bassa risoluzione.

Formato:

COLOR = *esprnm*

Tabella 8.2 — Codice dei colori in bassa risoluzione.

Cod.	**Colore**	**Cod.**	**Colore**	**Cod.**	**Colore**	**Cod.**	**Colore**
0	Nero	4	Verde scuro	8	Bruno	12	Verde chiaro
1	Magenta	5	Grigio # 1	9	Arancione	13	Giallo
2	Blu scuro	6	Blu medio	10	Grigio # 2	14	Acqua
3	Porpora	7	Blu chiaro	11	Rosa	15	Bianco

Sino alla successiva istruzione COLOR, le istruzioni PLOT, VLIN e HLIN saranno eseguite nel colore specificato.

Nella Tabella 8.2 riportiamo i codici dei colori. Il parametro *esprnm* deve avere un valore compreso tra 0 e 255; eventuali valori reali sono convertiti in intero. Per valori superiori a 15 si ripetono gli stessi codici (per esempio 0, 16, 32, ecc. rappresentano il nero; 1, 17, 33, ecc. rappresentano il magenta e così via). Il valore 0 (nero) viene imposto come default.

COLOR non opera in alta risoluzione. Se operate in modo-testo, COLOR è uno dei parametri che determinano quale carattere viene visualizzato con PLOT. Per una descrizione dettagliata vedere PLOT.

CON

Riporta in esecuzione un programma in Integer BASIC dopo una fermata.

Formato:

CON

CON riporta in esecuzione il programma dopo CTRL-C e alcune volte anche dopo RESET. Un programma non può continuare se fermato da CTRL-C durante una istruzione INPUT. Alcune volte CON può creare delle difficoltà se, dopo una fermata, vie-

ne cancellata o inserita una linea di programma o viene generato un messaggio di errore.

Può essere usata solo in modo immediato.

In Applesoft usate CONT.

CONT

Rimette in esecuzione un programma Applesoft dopo una fermata.

Formato:

CONT

CONT opera dopo che il programma è stato fermato con STOP, END o CTRL-C. Se una INPUT viene fermata con CTRL-C, il programma non può più essere continuato. CONT può generare l'errore ?CAN'T CONTINUE ERROR se, dopo una fermata, viene cancellata o inserita una linea di programma o viene dato un messaggio di errore.

In Integer BASIC usate CON.

DATA

Crea una lista di valori per essere assegnati mediante una READ.

Formato:

DATA *cost* [, *cost*...]

L'istruzione DATA può essere posta ovunque in un programma, anche dopo la corrispondente READ.

cost possono essere sia costanti numeriche che di stringa. Le stringhe possono non essere racchiuse tra virgolette salvo che la stringa contenga spazi, virgole e due punti. Un carattere virgolette non può essere contenuto in una *cost*; si deve ricorrere in tal caso alla funzione CHR$ (34) che rappresenta appunto le virgolette.

Una o più *cost* possono contenere solo spazi vuoti; in tal caso equivalgono a valori zero per le grandezze numeriche e a valori nulli per quelle di stringa.

Potete usare, senza errori, una DATA in modo immediato, ma essa *non* sarà accessibile da un comando READ.

DEF FN

Permette la definizione di funzioni dell'utente in Applesoft.

Formato:

DEF FN *nvar (arg)* = *esprnm*

La variabile reale *nvar* identifica la funzione che sarà poi richiamata con il nome FN *nvar.*

La funzione viene definita con l'espressione *esprnm. arg* è una variabile fittizia il cui nome può apparire (e ovviamente spesso appare) in *esprnm.* Il suo uso nella DEF FN non influenza assolutamente il valore di una eventuale altra variabile con lo stesso nome posta in altri punti del programma.

Quando FN *nvar* viene richiamata, il valore della variabile fittizia *arg* viene specificato con una espressione, o variabile, o costante numerica. Il valore di tutte le altre variabili che compaiono in *esprnm* devono essere definite prima che FN *nvar* sia usata. Vi rimandiamo anche a FN nella sezione Funzioni di questo capitolo.

Una istruzione DEF FN può occupare solo una linea. È però possibile citare altre funzioni di utente in una DEF FN, per cui si può costruire una funzione con la complessità che si vuole. Attenzione però che una funzione non può richiamare se stessa o un'altra funzione che a sua volta la richiami.

Una funzione definita dall'utente può essere ridefinita in uno stesso programma. Cioè si può usare lo stesso nome *nvar.*

DEF FN è disponibile solo in Applesoft.

Non è eseguibile in modo immediato, salvo che può essere richiamata da una FN, in modo immediato, prima di averla annullata con i comandi NEW, CLR o LOAD.

DEL

Elimina alcune linee di programma.

Formato:

DEL $linea_1$, $linea_2$

Vengono cancellate tutte le linee del programma in memoria dalla $linea_1$ alla $linea_2$. Se la $linea_1$ manca vengono cancellate tutte le linee da quella immediatamente superiore sino alla $linea_2$. Se manca la $linea_2$ vengono cancellate tutte le linee dalla $linea_1$ a quella immediatamente inferiore a $linea_2$.

DEL deve essere seguita da due numeri di linea separati da virgola che non possono essere negativi. Il secondo numero può essere maggiore o eguale al primo; in tal caso viene cancellata una sola linea.

DEL può essere usato solo in modo immediato in Integer BASIC. Se usato in modo differito (solo in Applesoft) vengono subito effettuate le cancellature e il programma si ferma. CONT non può però rimetterlo in esecuzione.

DELETE

Cancella un file da un disco.

Formato:

DELETE *nomefile* [, D*n*] [, S*n*] [, V*n*]

Il file *nomefile* viene cancellato dal disco.

Se il file non esiste nel drive D*n* dello slot S*n* viene dato l'errore FILE NOT FOUND. Se non vi è corrispondenza di volume V*n*, viene segnalato VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* o S*n* mancano, vengono imposti gli ultimi valori usati. V0 sostituisce V*n* se mancante. Se *n* manca vengono imposti i valori D0, S0 e V0.

DELETE è un comando DOS per cui richiede CTRL-D in una PRINT eseguita in modo differito.

DIM

Riserva spazio in memoria per le variabili con indici o le stringhe.

Questa istruzione richiede una descrizione separata tra Integer BASIC e Applesoft.

Formato in Integer BASIC:

DIM *var* (*ind*) [, *var* (*ind*) ...]

In Integer BASIC si possono dimensionare solo variabili numeriche con un indice o stringhe semplici. Lo spazio riservato in memoria è eguale al valore *ind* + 1. Attenzione che l'elemento 0 equivale alla stessa variabile senza indice (cioè A(0) = A).

Nel caso delle stringhe viene dichiarata la loro massima lunghezza (e non una struttura vettoriale). Ogni *ind* può essere tra 1 e 255. Il numero dei dimensionamenti consentiti dipende solo dall'ampiezza della memoria disponibile.

Se richiamate un elemento, di una variabile con indici, con un indice superiore al valore massimo dichiarato, vi viene segnalato l'errore * * * RANGE ERR. Se tentate di porre in una stringa più caratteri di quanti dichiarati, vi viene segnalato l'errore * * * STRING ERR.

DIM non effettua alcuna operazione iniziale di azzeramento sui suoi elementi numerici; dovete quindi fare voi l'azzeramento iniziale. Le stringhe vengono invece poste al valore nullo.

Formato in Applesoft:

DIM *var* (*ind* [, *ind* ...]) [, *var* (*ind* [, *ind* ...]) ...]

Le variabili possono avere uno o più indici:

$var(ind_i)$	variabile vettoriale
$var(ind_i, ind_j)$	variabile matriciale bidimensionale
$var(ind_i, ind_j, ind_k \ldots)$	variabile matriciale tridimensionale

L'Applesoft prevede tre tipi di variabili con indici: le intere, le reali e le stringhe. Ogni elemento di una variabile deve essere del tipo della variabile stessa. Il numero delle dimensioni, in una di queste variabili, dipende dal suo numero di indici. Quando un elemento viene richiamato, i suoi indici effettivi devono rientrare nei limiti dichiarati nella DIM

Il numero di dimensionamenti dipende dalla memoria disponibile. Il massimo numero di indici che una variabile può avere è 88, ma è possibile usarli solo se la maggior parte hanno valore 0. Se una variabile ha 89 indici o vengono superate le limitazioni di memoria, viene segnalato l'errore? OUT OF MEMORY ERROR!

Se tentate di usare un elemento con valori di indici non consentiti, vi viene segnalato l'errore ?BAD SUBSCRIT ERROR.

È possibile non dimensionare una variabile quando i suoi indici non superano il valore 10 ognuno (dimensionamento per default).

Una variabile non può essere dimensionata due o più volte anche se la prima volta è stata dimensionata per default. In caso contrario generate l'errore? REDIM'D ARRAY ERROR!

DRAW

Questa istruzione Applesoft permette di tracciare una figura in alta risoluzione.

Formato:

DRAW *esprnm* [AT *colh, righ*]

La figura specificata con il numero *esprnm* è tracciata con il colore imposto dall'ultima istruzione HCOLOR. I fattori di scala e di rotazione SCALE e ROT devono essere dati prima di DRAW.

La figura viene tracciata a partire dal punto indicato con le coordinate *colh* e *righ.* Se questi due parametri sono omessi, il tracciamento inizia dall'ultimo punto tracciato dall'ultima DRAW o XDRAW o PLOT.

Il numero di figura *esprnm* deve essere compreso tra 0 e il numero massimo di figure presenti nella tavola (che non può superare 255).

Non usate DRAW se non avete caricato in memoria una tavola delle figure. Potreste entrare in uno stato di errore o tracciare qualcosa assolutamente a caso!

DRAW non è disponibile in Integer BASIC.

DSP

Visualizza i cambiamenti della variabile specificata durante l'esecuzione di un programma in Integer BASIC.

Formato:

DSP *var*

Il valore della variabile *var* viene visualizzato, assieme al numero di linea, ogni volta che subisce un cambiamento. Attenzione che queste visualizzazioni interagiscono con le normali uscite del vostro programma rendendole forse illeggibili. RUN cancella tutte le istruzioni DSP. Usate CON o GOTO quando volete porre in esecuzione un programma e controllarlo in modo immediato con DSP.

Per annullare DSP, date NO DSP.

Non è disponibile in Applesoft.

END

Esegue la fine di un programma.

Formato:

END

Non appare alcun messaggio di fine. In Integer BASIC, END deve essere l'ultima istruzione eseguita; diversamente viene dato l'errore ***NO END ERR. END è facoltativa in Applesoft.

Non può essere usata in modo immediato in Integer BASIC.

EXEC

Esegue un file su disco come se ogni carattere del file fosse stato battuto alla tastiera.

Formato:

EXEC *nomefile* [, R*n*] [, D*n*] [, S*n*] [, V*n*]

Un file per essere usato con EXEC deve essere composto di comandi BASIC, di comandi DOS o di linee di programma. Quando EXEC viene eseguito preleva una linea dal file: se questa linea rappresenta un comando in modo immediato lo esegue subito, se invece rappresenta una istruzione differita la pone in memoria come se essa fosse stata battuta alla tastiera.

Con EXEC potete caricare un intero programma, listarlo, eseguirlo, salvarlo, cambiarlo o fare qualunque altra cosa che potreste fare alla tastiera. Potete anche creare ed eseguire un secondo file EXEC.

Il parametro R indica quale campo eseguire per primo. R conta dall'inizio del file: il primo campo è quello 0, il secondo è il campo 1, ecc. Il parametro di R deve essere un intero compreso tra 0 a 32767. Se R punta al primo campo oltre la fine del file non succede nulla. Se punta a due o più campi oltre la fine del file appare il messaggio END OF DATA.

Se una INPUT vien eseguita mentre un file EXEC è aperto, la risposta viene prelevata dal file EXEC.

Dopo l'esecuzione dell'ultima linea del file, il file si chiude da solo (vedere CLOSE).
Quando un comando EXEC viene eseguito all'interno di un altro file EXEC, il file originale viene chiuso ed ogni altro suo comando viene ignorato. Il nuovo file EXEC viene aperto e regolarmente eseguito.

Se il file non esiste sul disco nel drive *Dn* dello slot *Sn*, viene generato l'errore FILE NOT FOUND. Se non vi è corrispondenza di volume *Vn*, si genera l'errore VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* e S*n* sono omessi, vengono imposti gli ultimi valori usati. Se V*n* è assente viene imposto il valore V0. Se *n* manca vengono imposti i valori DO, SO e VO.

EXEC è un comando DOS e richiede CTRL-D in una PRINT in modo differito.

FLASH

Abilita il modo a intermittenza di visualizzazione sullo schermo.

Formato:

FLASH

Dopo l'esecuzione di FLASH, le istruzioni PRINT faranno visualizzare sullo schermo i caratteri con campo normale e campo invertito alternativamente. Anche i messaggi di errore subiranno lo stesso effetto. Solamente i caratteri battuti alla tastiera, dopo una INPUT (cioè l'eco), non subiranno l'effetto di FLASH.

Questo comando opera modificando il codice ASCII. Di conseguenza se inviate dei dati ad un disco, mentre FLASH è attivo, potreste ottenere in seguito una lettura del disco completamente errata.

Non è disponibile in Integer BASIC.

FN

Vedere più avanti la sezione riguardante le Funzioni e anche DEF FN.

FOR

Istruzione di ciclo che fa eseguire un certo numero di volte un gruppo di linee.

Formato:

FOR *varnm* = $esprnm_1$ TO $esprnm_2$ [STEP $esprnm_3$]

Quando l'istruzione FOR viene eseguita, alla variabile *varnm* viene assegnato il valore $esprnm_1$. Tutte le istruzioni successive a FOR vengono eseguite sino all'istruzione NEXT. Dopodichè *varnm* viene incrementata del valore $esprnm_3$ (oppure di 1 se

STEP è assente). Il nuovo valore di *varnm* viene confrontato con $esprnm_2$. Il senso del confronto dipende dal segno di $esprnm_3$. Le istruzioni comprese tra FOR e NEXT saranno ripetute varie volte sin tanto che *varnm* sia minore, o eguale, a $esprnm_2$ nel caso di $esprnm_3$ postitivo; nel caso invece di $esprnm_3$ negativo il ciclo sarà ripetuto sin tanto che *varnm* sia maggiore o eguale a $esprnm_2$. Siccome il confronto tra *varnm* e $esprnm_2$ viene eseguito dopo l'incremento, il ciclo viene sempre eseguito almeno una volta. Terminato il ciclo il programma prosegue all'istruzione successiva a NEXT.

In Integer BASIC *varnm* deve essere una variabile intera; in Applesoft può essere una variabile reale. In nessun caso può essere una stringa.

I valori di $esprnm_1$, $esprnm_2$ e $esprnm_3$ sono determinati una sola volta all'inizio del ciclo per cui non ha alcun effetto cambiarli durante l'esecuzione del ciclo. È possibile invece cambiare il valore *varnm* durante il ciclo; ciò è importante quando si vuole terminare anzitempo un ciclo per cui basta porre $varnm = esprnm_2$.

Non iniziate un ciclo all'esterno di una subroutine per terminarlo poi al suo interno.

Più cicli FOR-NEXT possono essere nidificati; ogni ciclo deve però avere una diversa variabile *varnm.* Ogni ciclo deve essere totalmente contenuto nel ciclo più esterno. Più cicli possono però terminare nello stesso punto. In Integer BASIC si possono avere sino a 16 livelli di nidificazione; in Applesoft invece sino a 10.

Il ciclo FOR può essere eseguito, in modo immediato, solo in Applesoft, ma l'intero ciclo deve essere battuto su una unica linea.

Se NEXT è omesso il ciclo viene eseguito una sola volta.

FP

Richiama l'interprete Applesoft.

Formato:

FP [, D*n*] [, S*n*] [, V*n*]

Il caricamento dell'Applesoft dipende da quale tipo di calcolatore Apple II avete:

1. L'Apple II Plus ha il linguaggio Applesoft nelle memorie ROM, indipendentemente da tutte le altre opzioni.
2. Se avete la cartolina firmware con l'Applesoft installata, FP carica il linguaggio da questa cartolina indipendentemente dalla posizione dei suoi interruttori.
3. Se è installata la cartolina Apple Language System, FP carica da questa cartolina.
4. In tutti gli altri casi FP ricerca l'Applesoft sul disco in quel momento di default. Se non lo trova invia il messaggio LANGUAGE NOT AVAILABLE.

FP cancella ogni programma già presente in memoria. Se non vi è corrispondenza

dei parametri D*n*, S*n* e V*n* si generano gli errori FILE NOT FOUND E VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* e S*n* mancano vengono imposti gli ultimi valori usati. V0 è usato nel caso di V*n* mancante. Se *n* non viene specificato, si impone D0, S0 e V0.

FP opera solo in modo immediato.

GET

Questo comando Applesoft riceve un solo carattere dalla tastiera senza farne l'eco sullo schermo.

Formato:

GET *var*

L'esecuzione attende che venga battuto un tasto. Se battete CTRL-@ viene assegnata a *var* la stringa nulla.

GET non viene normalmente usata per variabili numeriche. Se la usate con variabili numeriche non avrete problemi sintantochè le assocerete i valori numerici 0, 1, ... 9, ma se battete un segno più o meno, una virgola, due punti, CTRL-@, spazio, E o punto viene assegnato invece il valore zero alla variabile. Per qualunque altro carattere battuto si genera l'errore? SYNTAX error e il programma si ferma.

GET non può essere usata in modo immediato.

Non è disponibile in Integer BASIC.

GOSUB

Esegue il salto ad una subroutine. Quando nella subroutine viene incontrata una istruzione RETURN il programma ritorna all'istruzione subito successiva a GOSUB.

Formato generale:

GOSUB *linea*

La subroutine deve iniziare in senso logico alla linea *linea;* ciò significa che possono esistere nella subroutine linee con numerazione inferiore. Al termine logico della subroutine deve essere posta una istruzione RETURN per il ritorno al programma principale.

Le istruzioni GOSUB possono essere poste ovunque in un programma.

Più subroutine possono essere nidificate; cioè una subroutine può chiamare un'altra subroutine. In Applesoft sono possibili 25 livelli di nidificazione. Ciò significa che si possono eseguire 24 livelli di istruzione GOSUB prima che si esegua una RETURN. In Integer BASIC si possono avere invece solamente 16 GOSUB.

Da una subroutine si deve uscire normalmente mediante la sua RETURN e non con GOTO. È possibile però usare GOTO se prima si è eseguita una istruzione POP.

In Integer BASIC, GOSUB non può essere usata in modo immediato.

Formato specifico per l'Integer BASIC:

GOSUB *esprnm*

In Integer BASIC si può porre una espressione numerica per indicare il numero di linea. Se *esprnm* non porta ad un numero di linea esistente si genera l'errore ***BAD BRANCH ERR. In questo modo è possibile simulare l'istruzione tipo ON-GOSUB che non esiste in Integer BASIC.

GOTO

È l'istruzione di salto incondizionato.

Formato generale:

GOTO *linea*

Il programma prosegue alla linea indicata. Se la linea *linea* manca si genera l'errore? UNDEF'D STATEMENT ERROR in Applesoft e l'errore ***BAD BRANCH ERR in Integer BASIC.

Formato specifico per l'Integer BASIC:

GOTO *esprnm*

In Integer BASIC si può porre una espressione numerica per indicare il numero di linea. Se *esprnm* non porta ad un numero di linea esistente si genera l'errore ***BAD BRANCH ERR. In questo modo è possibile simulare l'istruzione ON-GOTO che non esiste in Integer BASIC.

GR

Converte lo schermo nel modo grafico a bassa risoluzione (40 × 40) e lascia quattro linee in basso disponibili per un testo.

Formato:

GR

La porzione grafica dello schermo viene portata a sfondo nero mentre il cursore va all'inizio della finestra di testo. COLOR viene posto a 0 (nero).

Se GR viene eseguita mentre HGR è in funzione, essa si comporta normalmente.

Se invece è in funzione HGR2, rimarrete in collegamento con la pagina 2 dei grafici e del testo. Questo può portare a della confusione perchè lo schermo si riempirà di caratteri qualunque, mentre ciò che batterete alla tastiera non avrà eco sullo schermo. Per ritornare al modo normale battete TEXT. Assicuratevi di usare TEXT nei vostri programmi prima di passare da HGR2 a GR.

Mediante il comando POKE −16302,0 potete passare allo schermo grafico completo (40 × 48) in bassa risoluzione, dopo aver eseguito GR. Se ora battete delle istruzioni in modo immediato, vi appariranno sullo schermo in basso come un insieme di punti colorati, ma saranno eseguite correttamente. Per ripristinare la finestra di testo battete POKE −16301,0.

HCOLOR =

Questa istruzione Applesoft impone i colori per i grafici ad alta risoluzione.

Formato:

HCOLOR = *esprnm*

HCOLOR specifica un colore che rimane in atto sino alla prossima HCOLOR. Nella Tabella 8.3 sono riportati i codici dei colori in alta risoluzione. *esprnm* deve essere compreso tra 0 e 7. I valori fuori da questo intervallo causano l'errore? ILLEGAL QUANTITY ERROR. Se una istruzione grafica in alta risoluzione viene data prima di una HCOLOR, essa sarà seguita con un colore indeterminato.

HCOLOR non influenza i grafici a bassa risoluzione. Una istruzione HCOLOR, eseguita mentre il calcolatore non è nel modo grafico ad alta risoluzione, non ha influenza sul colore dei successivi grafici in alta risoluzione.

Non esiste in Integer BASIC.

Tabella 8.3 — Codice dei colori in alta risoluzione

Codice	Colore	Codice	Colore
0	Nero	4	Nero
1	Verde	5	Arancione *
2	Violetto *	6	Blu *
3	Bianco	7	Bianco
* Dipende dalla regolazione del televisore			

HGR

Questa istruzione Applesoft converte lo schermo nel modo grafico ad alta risoluzione (280 × 160) con quattro linee in basso per i testi.

Formato:

HGR

Viene visualizzata la pagina 1 ad alta risoluzione. La memoria dello schermo a bassa risoluzione (per testi) non viene modificata, ma rimangono visibili solo le quattro linee in basso. Il cursore non appare in queste linee e in genere bisogna battere molte linee prima di vederlo apparire. La parte grafica dello schermo viene portata in nero. HCOLOR non risente del comando HGR.

È possibile passare allo schermo grafico completo (280 × 192), in alta risoluzione, mediante il comando POKE −16302,0 dopo l'esecuzione di HGR. Ogni comando che darete successivamente in modo immediato non sarà visibile, ma perfettamente eseguibile. Con POKE−16301,0 potete ripristinare la finestra per il testo.

Se il calcolatore Apple II ha meno di 32K di memoria, non potete usare contemporaneamente il DOS e il comando HGR perchè userebbero la stessa zona di memoria.

Inoltre l'interprete Applesoft, proveniente da disco o da cassetta, occupa parte della memoria riservata per la pagina grafica 1 ad alta risoluzione. Non potete quindi usare HGR con l'Applesoft proveniente da disco o da cassetta.

Anche nel caso dell'Applesoft su firmware, se il vostro programma è molto lungo può sovrapporsi alla pagina 1. Potete ovviare a questo inconveniente con l'istruzione HIMEM: 8192 che pone il vostro programma fuori della pagina 1.

HGR non è disponibile in Integer BASIC.

HGR2

Converte lo schermo in modo grafico ad alta risoluzione (280 × 192) e visualizza la pagina 2 ad alta risoluzione.

Formato:

HGR2

La memoria dello schermo a bassa risoluzione (per testi) non viene influenzata. Anche se non potete vedere ciò che battete, ogni comando battuto viene regolarmente eseguito. Lo schermo viene portato nel colore nero. HCOLOR non risente del comando HGR2.

La pagina 2 non è disponibile se avete meno di 24K di memoria. Per un sistema con almeno 24K ponete HIMEM: al valore 16384, prima di usare HGR2, per proteggere sia il vostro programma che le sue variabili e viceversa i grafici.

Per usare assieme il DOS e il comando HGR2 dovete avere almeno 36K di memoria. Ciò significa che dovete avere almeno 36K per potere usare HGR2 con l'Applesoft proveniente da disco!

Non tentate di aprire la finestra per il testo con POKE –16301,0. In tal caso i vostri comandi in modo immediato andrebbero sulla pagina 1 e sarebbero quindi invisibili (anche se vengono eseguiti correttamente).

HGR2 non è disponibile in Integer BASIC.

HIMEM:

Fissa un limite superiore alla memoria disponibile per i programmi in BASIC e per l'area delle variabili.

Formato:

HIMEM: *esprnm*

HIMEM: stabilisce l'indirizzo più alto della memoria RAM disponibile per i programmi BASIC. Il sistema operativo a disco DOS risiede sempre sopra HIMEM: (se presente). Con questo comando potete riservare ulteriore spazio in memoria per i vostri programmi in assembler e per le tavole grafiche in alta risoluzione. Potete anche proteggere la memoria RAM dello schermo per grafici in alta risoluzione.

HIMEM: viene inizialmente posto al valore più alto della vostra memoria (per esempio 49151 per un Apple II con 48K). Il DOS risiede nella parte alta della memoria per cui sposta HIMEM: di 10.800 byte circa più in basso quando viene caricato. Ogni ulteriore buffer, definito con MAXFILES, occupa altri 595 byte, per cui si abbassa ancora HIMEM:. Se ora il vostro programma usa delle stringhe esse saranno memorizzate a partire da quest'ultimo valore di HIMEM: verso il basso. Vedere la mappa di memoria dell'Appendice G.

esprnm deve essere compreso nell'intervallo tra -65535 e 65535 (-32767 e 32767 in Integer BASIC); diversamente si genera un errore.

Non potete porre HIMEM: ad un valore superiore alla massima memoria che avete disponibile. In tal caso potreste assegnare a delle variabili indirizzi non esistenti.

Potete leggere il valore corrente di HIMEM: mediante queste istruzioni:

PRINT PEEK (116) * 256 + PEEK (115)	per l'Applesoft
PRINT PEEK (77) * 256 + PEEK (76)	per l'Integer BASIC

HIMEM: non è influenzato da NEW, RUN, e CLEAR.

Se ponete HIMEM: inferiore a LOMEM: o non lasciate sufficiente spazio per il vostro programma, generate un errore.

In modo immediato può essere usato solo in Integer BASIC.

HLIN

Traccia una linea orizzontale sullo schermo nel modo grafico a bassa risoluzione.

Formato:

HLIN *col*$_1$, *col*$_2$ AT *rig*

Viene tracciata una linea sulla riga indicata tra le due colonne specificate. Il colore è determinato dall'ultima istruzione COLOR eseguita. Se lo schermo è in modo testo, o è presente la finestra di testo e *rig* è superiore a 39, HLIN traccerà una linea di caratteri sullo schermo invece dei punti grafici. I caratteri visualizzati sono determinati dalla istruzione COLOR precedente; leggete più avanti l'istruzione PLOT e la Tabella 8.5 per maggiori particolari.

In Integer BASIC *col*$_1$ deve essere minore o eguale di *col*$_2$ altrimenti appare il messaggio * * * RANGE ERR.

HOME

In Applesoft viene pulito lo schermo e il cursore viene portato in alto a sinistra (riga 1 e colonna 1).

Formato:

HOME

In Integer BASIC dovete usare CALL -936.

HPLOT

In Applesoft questa istruzione traccia un punto o una linea di colore sullo schermo in alta risoluzione.

Formato:

HPLOT *colh, righ*
HPLOT TO *colh, righ*
HPLOT *colh*$_1$, *righ*$_1$ TO *col*$_2$, *righ*$_2$ [TO *colh*$_3$, *righ*$_3$...]

Il primo formato fa tracciare un punto alle coordinate indicate. Il colore dipende dall'ultima istruzione HCOLOR eseguita.

Il secondo formato fa tracciare una linea dall'ultimo punto già tracciato sino al punto di coordinate indicate *colh* e *righ*. Se manca un punto già tracciato, dopo HGR o HGR2, l'istruzione non ha effetto. Il colore è determinato dall'ultima istruzione HCOLOR.

Anche il terzo formato fa tracciare una linea in colore. Con l'Applesoft su firmware la linea può avere però più segmenti. Tutti i segmenti sono sempre tracciati nel colore determinato dall'ultima HCOLOR eseguita. La linea viene dapprima tracciata tra

colh$_1$/*righ*$_1$ e *colh*$_2$/*righ*$_2$. Poi se presente l'Applesoft su firmware, vengono tracciati gli altri segmenti tra *colh*$_2$/*righ*$_2$ e *colh*$_3$/*righ*$_3$ e così avanti.

Nel caso di Applesoft proveniente da cassetta o da disco non è possibile tracciare gli altri segmenti.

Il numero di segmenti consentiti dipende solo dalla limitazione di inserire i loro estremi in una linea di programma.

La parte di tracciato che entra nella finestra di testo non sarà visibile. Se togliete però la finestra di testo con POKE −16302,0 anche questa parte di linea diverrà visibile.

Prima di HPLOT dovete sempre eseguire HGR o HGR2, altrimenti potete distruggere il vostro programma o le sue variabili.

HPLOT non è disponibile in Integer BASIC.

HTAB

Questa istruzione Applesoft posiziona il cursore alla colonna indicata sulla riga corrente.

Formato:

HTAB *col*

Il cursore si porta sulla colonna indicata senza cancellare alcun carattere. Le colonne sono numerate da 1 a 40 da sinistra verso destra.

In Integer BASIC usate l'istruzione TAB.

IF-THEN

È l'istruzione di esecuzione condizionale. Le regole per l'Integer BASIC e l'Applesoft sono presentate separatamente.

Formato in Integer BASIC:

IF *espr* THEN *istruzione*
IF *espr* THEN [GOTO] *linea*

Nel primo caso se la condizione espressa da *espr* si verifica, viene eseguita *l'istruzione*. Se invece la condizione è falsa viene eseguita l'istruzione che segue tutta la IF-THEN e non viene eseguita l'*istruzione* che compare in IF-THEN.

Il secondo formato rappresenta un vero e proprio salto condizionale. Se la condizione è verificata il programma prosegue alla *linea* indicata.

Il tipo più comune di espressione usato è quello relazionale. Valori di stringa possono essere solo confrontati per l'eguaglianza o la diseguaglianza in Integer BASIC.

espr può essere anche una espressione numerica. In questo caso *espr* è considerata vera se ha un valore non nullo.

L'espressione di IF-THEN non può essere una espressione di stringa (o qualcosa riconducibile ad un valore di stringa) in Integer BASIC.

Formato in Applesoft:

IF *espr* THEN *istruzione:* [: *istruzione...*]

IF espr { THEN / GOTO / THEN GOTO } *linea*

Nel caso del primo formato se la condizione *espr* risulta vera, vengono eseguite tutte le *istruzioni* che seguono THEN sulla stessa linea. Se la condizione è falsa viene eseguita l'istruzione che si trova sulla linea successiva alla IF-THEN e non vengono eseguite le *istruzioni* che seguono THEN.

Con il secondo formato si ha una istruzione di salto condizionale. Se la condizione è vera il programma prosegue alla *linea* indicata. Altrimenti il programma prosegue con le istruzioni poste sulla linea di programma subito successiva alla IF-THEN.

Se una delle *istruzioni* che seguono THEN è un salto incondizionato, allora tale istruzione deve essere della forma GOTO *linea.* Tale istruzione GOTO deve essere ovviamente l'ultima perchè le successive non sarebbero mai eseguite in nessun caso.

Il tipo più comune di *espr* usato in IF-THEN è l'espressione relazionale. Se vengono eseguiti confronti tra valori di stringa, il confronto viene in effetti ricondotto a quello tra i valori del codice ASCII dei singoli caratteri (vedere l'Appendice I). Il confronto fra stringhe avviene carattere per carattere. A parità di caratteri di sinistra, viene considerata maggiore la stringa più lunga.

espr può essere anche numerica e in tal caso è considerata vera se assume un valore diverso da zero; mentre è considerata falsa se assume il valore zero.

In Applesoft *espr* può essere anche una espressione di stringa. Se però usate più di due o tre di queste IF-THEN in un programma, è probabile che il calcolatore vi segnali l'errore? FORMULA TOO COMPLEX ERROR.

In Applesoft dovete stare attenti che il carattere, non spazio, che precede la T di THEN non sia una A perchè in tal caso l'interprete BASIC riconoscerebbe la parola riservata AT. Per evitare ciò è sufficiente porre tra parentesi parte o tutta la *espr.*

Se un ciclo FOR-NEXT segue la THEN è allora necessario che tutte le istruzioni del ciclo siano contenute sulla stessa linea della IF-THEN. Analogamente altre istruzioni IF-THEN possono seguire una THEN, ma devono essere tutte contenute sulla stessa linea. Talvolta è più opportuno usare degli operatori Booleani invece di porre più IF-THEN in successione. Nell'esempio che segue si vede chiaramente che è preferibile la secondo IF-THEN rispetto alla prima:

```
10 IF A$ = "X" THEN IF B = 2 THEN IF C = 6 THEN 50
10 IF A$ = "X" AND B = 2 AND C = 6 THEN 50
```

IN

Seleziona lo "slot" della periferica da cui saranno ricevuti i successivi dati d'ingresso.

Formato:

IN# *slot*

Le istruzioni INPUT successive riceveranno i dati dalla periferica collegata allo slot indicato. *slot* deve essere un valore intero compreso tra 0 e 7. Il valore 0 è riservato però per la tastiera per cui IN# 0 indica la tastiera standard. Se nessuna periferica è collegata allo slot indicato, il calcolatore entra in una fase di attesa che può essere interrotta solo da RESET.

Se il DOS è stato caricato, IN# viene considerato un comando DOS per cui richiede di essere posto in una PRINT con CTRL-D per il modo differito.

INIT

Inizializza un disco.

Formato:

INIT *nomefile* [, D*n*] [, S*n*] [, V*n*]

Con questo comando il programma attualmente presente nella memoria RAM viene portato sul disco con il nome *nomefile.* Tale programma diviene il programma iniziale del disco e viene sempre caricato in memoria ogni volta che viene fatto il booting del disco. Il disco riceve il numero di volume V*n.* Se tale parametro non viene specificato, viene imposto il valore implicito 254.

D*n,* S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* e S*n* sono omessi vengono imposti gli ultimi valori usati. Se *n* manca vengono imposti i valori D0 e S0.

INIT può essere usato solo in modo immediato.

INPUT

Accetta caratteri in ingresso dalla tastiera o da un'altra periferica d'ingresso e li assegna alle variabili della lista specificata.

Formato in Integer BASIC:

INPUT ["*avviso*",] *var* [, *var* ...]

INPUT richiede in ingresso dei valori per ogni sua variabile sia intera che di stringa. Se la prima variabile è intera allora appare un punto interrogativo, nella posizione

attuale del cursore per avvisare che devono essere dati dei valori in ingresso. Se la prima variabile è di stringa l'Integer BASIC non visualizza il punto interrogativo.

"*avviso*" è una stringa che viene usata come messaggio per l'operazione di ingresso. Essa viene visualizzata una sola volta seguita dal punto interrogativo nel caso la prima variabile sia intera. Anche in questo caso manca il punto interrogativo se la prima variabile è di stringa. Notate che la stringa "*avviso*" è seguita da una virgola. "*avviso*" non può essere nè una variabile nè una espressione.

Se dovete dare in ingresso valori interi, questi possono essere dati su una stessa linea separati da virgole, oppure su linee diverse separati dal ritorno del carrello. L'Integer BASIC fa apparire due punti interrogativi (??) su ogni nuova linea per ricordare che devono essere dati altri valori. Possono essere dati anche più valori su una stessa linea separati da virgole.

I valori numerici possono essere costituiti dalle dieci cifre 0-9, dai segni più e meno o dallo spazio. Se date RETURN, senza battere alcuna cifra, vi viene segnalato errore.

Ogni valore di stringa deve essere battuto su una linea diversa. Tutti i caratteri che battete (esclusi CTRC-C e CTRL-X) prima di RETURN verranno a far parte della stringa. Il carattere nullo ("") viene assegnato anche solo premendo RETURN.

Se battete dei caratteri non accettabili, come per esempio le lettere per una variabile numerica, vi vengono dati i messaggi * * * SYNTAX ERR e RETYPE LINE. In tal caso dovete ribattere tutti i caratteri della linea errata.

INPUT non può essere usato in modo immediato.

Formato in Applesoft:

INPUT ["*avviso*";] *var* [, *var* ...]

INPUT richiede dei valori in ingresso per ogni sua variabile sia intera che di stringa. Un punto interrogativo (?) viene normalmente visualizzato per indicare che devono essere dati dei valori in ingresso. Tale punto interrogativo appare nella posizione del cursore. In Applesoft il punto interrogativo non appare se viene usato il messaggio opzionale "*avviso*".

"*avviso*" è una costante di stringa. Se presente viene visualizzato subito prima del primo valore in ingresso e non viene ripetuto per le altre variabili della lista. Dopo l'eventuale "*avviso*" non compare il punto interrogativo. Notate che in Applesoft l'"*avviso*" è seguito da punto e virgola.

È possibile dare più valori su una stessa linea separandoli con virgole. Altrimenti si può andare a capo più volte con il ritorno del carrello.

Se battete dei caratteri non accettabili per la variabile associata, come le lettere al posto delle cifre per una variabile numerica, compare il messaggio REENTER e dovete quindi rieseguire tutta la INPUT. L'"*avviso*" o il punto interrogativo vengono nuovamente visualizzati e dovete allora ribattere tutti i dati sin dall'inizio.

I valori numerici possono essere costituiti dalle cifre 0-9, dai segni più e meno, dal-

lo spazio e inoltre dal punto decimale e dalla lettere E (per la notazione scientifica dei numeri). Non potete battere solo RETURN per il valore 0.

In Applesoft se il primo carattere, non spazio, per una stringa è quello di virgolette, tutti caratteri successivi (compresi le virgole e i due punti) saranno inclusi nella stringa sino alle prossime virgolette o a RETURN. Se invece il primo carattere è diverso dalle virgolette, tutti i caratteri successivi (comprese le virgolette) saranno inclusi nella stringa sino alla prossima virgola o due punti o RETURN. Se una INPUT richiede due o più stringhe, allora esse devono essere poste tra virgolette e separate da virgole.

Premere solo RETURN equivale a dare la stringa nulla ("").

In Applesoft tutti i caratteri, dopo una virgola nella risposta ad una INPUT, sono trascurati a meno che il primo carattere battuto sia quello di virgolette.

INPUT non può essere usato in modo immediato.

INT

Commuta il calcolatore in Integer BASIC.

Formato:

    INT

Ogni programma in memoria viene cancellato. Se l'Integer BASIC non è disponibile (per esempio un Apple II Plus senza Language System), verrà visualizzato il messaggio LANGUAGE NOT AVAILABLE.

Può essere usato solo in modo immediato.

INVERSE

È una istruzione Applesoft che permette di invertire il campo dei caratteri sullo schermo.

Formato:

    INVERSE

Tutte le successive uscite (PRINT) saranno visualizzate in campo inverso. Anche i messaggi di errore saranno in campo inverso. Solo i caratteri in ingresso, a seguito di una INPUT, non saranno invertiti.

INVERSE opera modificando il codice ASCII dei caratteri per cui se registrate su disco dei caratteri invertiti, otterrete sicuramente degli errori nella lettura successiva.

INVERSE funziona solo in Applesoft.

LET =

L'istruzione LET =, o più semplicemente =, assegna un valore ad una variabile.

Formato:

[LET] *var* = *espr*

Alla variabile *var* viene assegnato il valore ottenuto dal calcolo dell'espressione *espr.*

LIST

Visualizza in parte o tutto il programma presente nella memoria.
Esistono due formati di questa istruzione. Il primo è riconosciuto sia dal BASIC che dall'Applesoft, il secondo solo dall'Applesoft.

Formato generale:

LIST $linea_1$ [, $linea_2$]

Viene listato il programma dalla $linea_1$ alla $linea_2$. Se i due parametri di linea mancano viene listato tutto il programmma. Se è presente solo $linea_1$ viene listata solo quella linea.
Se la $linea_1$ non esiste nel programma, il listato inizia da quella immediatamente superiore; se invece manca la $linea_2$, il listato termina a quella immediatamente inferiore.

LIST non può essere usato con variabili o espressioni al posto dei numeri di linea.
LIST inserisce automaticamente spaziature intorno alle variabili e alle parole riservate per rendere il testo più leggibile. Potete togliere queste spaziature con il comando POKE 33,33; con POKE 33,40 ripristinate le condizioni precedenti. Se battete una linea senza spaziature potete aumentare la sua lunghezza apparente, ma essa può risultare troppo lunga quando la listate con l'inserimento delle spaziature.

Formato aggiuntivo per l'Applesoft:

$$\text{LIST}\begin{cases} linea_1 & [\left\{\begin{matrix}-\\,\end{matrix}\right\}] \\ [linea_1] & \left\{\begin{matrix}-\\,\end{matrix}\right\} \quad linea_2 \end{cases}$$

In Applesoft sia una virgola che un trattino possono separare i due numeri di linea.
In Applesoft se manca $linea_1$ il programma viene listato dall'inizio sino alla $linea_2$ (con la virgola o il trattino presenti). Analogamente se manca $linea_2$, ma sono presenti o la virgola o il trattino, il programma viene listato da $linea_1$ sino alla fine.

LOAD

Carica un programma da una cassetta o da un disco.

Formato per le cassette:

LOAD

Carica il programma successivo dalla cassetta, sostituendo ogni altro programma nella memoria RAM. Il registratore deve già essere in PLAY al momento dell'esecuzione di LOAD. Ricordatevi che il calcolatore non vi dà alcun avviso di premere questo tasto. Il calcolatore produce però due "beep" all'inizio e alla fine del caricamento. Dopo il secondo "beep" potete fermare il registratore manualmente.

LOAD può essere usato in modo immediato solo in Integer BASIC.

Formato per i dischetti:

LOAD *nomefile* [, D*n*] [, S*n*] [, V*n*]

Il programma con il nome *nomefile* è caricato dal dischetto. Ogni altro programma già presente in memoria viene cancellato.

Se il programma che deve essere caricato è in Applesoft e il calcolatore in Integer BASIC, o viceversa, l'Apple II cambia automaticamente nel linguaggio richiesto dal programma. Se il nuovo linguaggio non è disponibile viene dato il messaggio LANGUAGE NOT AVAILABLE.

Se il programma cercato non è presente sul disco nel drive D*n* dello slot S*n*, viene dato l'errore FILE NOT FOUND. Se non vi è corrispondenza nel numero di volume V*n*, viene dato l'errore VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* e S*n* mancano vengono imposti i valori degli ultimi drive e slot usati. Se V*n* è mancante viene imposto il valore VO. Se è mancante *n* vengono imposti i valori DO, SO e VO.

LOAD è un comando DOS e richiede quindi CTRL-D in una PRINT in modo differito.

LOCK

Protegge un file su disco da cancellature accidentali.

Formato:

LOCK *nomefile* [, D*n*] [, S*n*] [, V*n*]

Un file *protetto* (*locked*) non può essere nè cancellato nè cambiato di nome sino a che non venga tolta la protezione (vedere UNLOCK). Nessun programma può essere salvato con il nome di un file protetto. Un file protetto viene indicato nel catalogo del disco con un asterisco posto a sinistra.

Se il file non esiste sul disco nel drive D*n* dello slot S*n*, viene visualizzato il messaggio FILE NOT FOUND. Se non vi è corrispondenza nel numero di volume V*n*, viene dato l'errore VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* e S*n* mancano vengono imposti i valori degli ultimi drive e slot usati. Se V*n* è mancante viene imposto il valore V0. Se manca *n* vengono imposti i valori D0, S0 e V0.

LOCK è un comando DOS e richiede quindi CTRL-D in una PRINT in modo differito.

LOMEM:

Stabilisce il limite inferiore della zona di memoria riservata da un programma BASIC alle variabili.

Formato:

LOMEM: *esprnm*

LOMEM: individua l'indirizzo più basso della memoria RAM disponibile per le linee di un programma BASIC e per le variabili. Il Monitor e l'interprete BASIC usano la memoria sotto LOMEM: per i puntatori, per la grafica a bassa risoluzione, per i testi sullo schermo e per altre cose. Quando l'interprete Applesoft viene caricato da disco o da cassetta, esso risiede sempre sotto LOMEM:. Spostando in alto LOMEM: si possono riservare degli spazi aggiuntivi per le subroutine in linguaggio macchina e per le tavole delle forme in alta risoluzione.

In Integer BASIC, o con l'Applesoft su firmware, LOMEM: inizia dall'indirizzo 2048 appena sopra l'area di sistema. Se invece l'Applesoft viene caricato da disco o da cassetta LOMEM: viene portato a 12291. Ogni volta che scrivete una nuova linea di programma in Applesoft, o ne cambiate una già esistente, LOMEM: viene aggiustato in alto o in basso. Se cancellate un programma Applesoft (con NEW o CTRL-B) anche LOMEM: cambia. Se volete quindi riservare un'area di memoria sotto un programma dovete farlo dopo avere cancellato i vecchi programmi, ma prima di scrivere quello nuovo.

Il valore di *esprnm* deve essere compreso tra -65535 e 65535 (in Integer BASIC tra -32767 e 32767).

Potete visualizzare il valore corrente di LOMEM: con l'istruzione PRINT PEEK (106) * 256 + PEEK (105).

In Applesoft se LOMEM: è posto sopra a HIMEM:, oppure più in basso del valore già presente per LOMEM:, oppure più in basso del massimo indirizzo usato dal sistema operativo o dal programma, allora viene dato il messaggio ?OUT MEMORY ERROR.

Può essere usato in modo immediato solo in Integer BASIC.

MAN

Fa terminare la numerazione automatica delle linee in Integer BASIC.

Formato:

MAN

Con il comando AUTO si ha la numerazione automatica delle linee in Integer BASIC; con MAN si ritorna alla numerazione manuale.

Battete CTRL-X per fermare temporaneamente la numerazione automatica e poi battete MAN.

Non è disponibile in Applesoft.

MAXFILES

Indica il numero massimo di file su disco contemporaneamente aperti.

Formato:

MAXFILES *limite*

Il sistema operativo a disco DOS supporta sino a 16 file contemporaneamente aperti. Quando eseguite MAXFILES riservate un ulteriore buffer di 595 byte per ogni file. MAXFILE è posto automaticamente a 3 quando viene caricato il DOS.

Tutti i comandi DOS, eccetto MAXFILES, usano un buffer per potere operare. Così in pratica il numero di file massimo che potete aprire assieme, è uno di meno del limite imposto da MAXFILES. Se tentate di eseguire un comando DOS, quando non vi siano buffer disponibili, vi viene segnalato l'errore NO BUFFERS AVAILABLE.

MAXFILES sposta HIMEM: per cui può essere cancellato un vostro programma o le sue variabili. Cercate quindi di eseguire MAXFILES prima di caricare, ed eseguire, i programmi.

Se usate MAXFILES in un programma Applesoft, ponetelo nella prima linea. Per usarlo invece in Integer BASIC, dovete creare un file EXEC come vi abbiamo indicato nel capitolo 5 (vedere anche EXEC in questo capitolo).

limite deve essere una costante intera tra 1 e 16; in caso contrario vi viene segnalato l'errore RANGE ERROR.

MAXFILES è un comando DOS che richiede quindi CTRL-D in una PRINT in modo differito.

MON

Fa visualizzare sullo schermo i comandi al disco e/o i dati.

Formato:

MON [C] [, I] [, O]

I tre parametri stabiliscono ciò che verrà visualizzato sullo schermo. C fa visualizzare i comandi al disco. I fa visualizzare tutti i dati provenienti dal disco verso il calcolatore. O fa visualizzare i dati che vanno sul disco. I tre parametri possono essere dati singolarmente e con qualunque ordine. Se mancano tutti e tre, MON non ha effetto. MON rimane attivo sino a chè non vengano dati NOMON, o FP, o INT, oppure il sistema sia ricaricato, o venga dato RESET (solo per alcune macchine).

MON è un comando DOS e richiede CTRL-D in una PRINT in modo differito.

NEW

Cancella il programma corrente e le sue variabili dalla memoria RAM.

Formato:

NEW

NEW ripristina anche LOMEM:, ma non tocca HIMEM:, COLOR e HCOLOR.

NEW può essere usato in modo immediato solo in Integer BASIC.

NEXT

Termina un ciclo iniziato con l'istruzione FOR.

Formato generale:

NEXT *varnm* [, *varnm* ...]

Quando NEXT viene eseguita, la variabile di ciclo *varnm* è incrementata per il valore indicato da STEP. Viene quindi fatto un confronto, in accordo con i parametri dell'istruzione FOR, e se il ciclo è terminato il programma prosegue all'istruzione successiva alla NEXT.

Se non vi è una corrispondente istruzione FOR, o non vi è accordo per il nome della variabile di ciclo *varnm,* allora viene dato il messaggio ?NEXT WITHOUT FOR ERROR, in Applesoft, e * * * BAD NEXT ERR in Integer BASIC.

Più variabili possono essere poste dopo NEXT, ma nell'ordine opportuno. La prima *varnm* deve riferirsi al ciclo più interno.

NEXT non può essere usata in modo immediato in Integer BASIC. In Applesoft NEXT può essere usata in modo immediato, ma chiude un ciclo già presente in modo differito e tuttora aperto.

Ulteriore formato per l'Applesoft:

NEXT

In Applesoft si può prescindere dal nome della variabile di ciclo. Per default viene usata quella che si riferisce al ciclo aperto per ultimo. NEXT senza variabile viene eseguita in minor tempo che se vi fosse indicata la variabile *varnm.*

NO DSP

Annulla gli effetti di DSP (vedere DSP).

Formato:

NO DSP *var*

Disponibile solo in Integer BASIC.

NOMON

Fa terminare la visualizzazione dei comandi e dei dati iniziata con MON.

Formato:

NOMON [C] [, I] [, O]

Ogni parametro indicato annulla il corrispondente effetto dato con MON. Se viene indicato C non sono più visualizzati i comandi al disco. Se viene indicato I non sono più visualizzati i dati provenienti dal disco. Se viene specificato O non sono più visualizzati i dati in uscita verso il disco. I parametri possono essere dati in ogni combinazione e con qualunque ordine. Se MON non è attivo, NOMON non ha effetto.

NOMON è un comando DOS e richiede CTRL-D in una PRINT in modo differito.

NORMAL

Questa istruzione Applesoft annulla gli effetti di FLASH e INVERSE.

Formato:

NORMAL

Vedere FLASH e INVERSE. Non è disponibile in Integer BASIC.

NO TRACE

Annulla gli effetti dell'istruzione TRACE.

Formato:

NO TRACE

Se TRACE non è attiva, NO TRACE non ha effetto.

ONERR GOTO

Dopo questa istruzione, in un programma Applesoft, il programma prosegue alla linea indicata in caso di errore.

Formato:

ONERR GOTO *linea*

Questa istruzione pone un flag che farà saltare alla *linea* indicata in caso di errore. ONERR GOTO deve essere eseguita prima che si commetta l'errore.

Ogni tipo di errore ha un suo numero in codice. Il codice dell'errore commesso per ultimo è memorizzato all'indirizzo 222. Con PEEK (222) potete leggere il tipo di errore. Tutti i codici di errore sono riportati nella Tabella C.1 dell'Appendice C.

Quando un errore si verifica all'interno di un ciclo FOR-NEXT o in una subroutine i puntatori e gli stack vengono rovinati. La vostra routine di gestione dell'errore deve quindi provvedere a rieseguire il ciclo o la subroutine.

In alcuni casi ONERR GOTO non funziona correttamente. Il calcolatore Apple II si blocca se trova due errori di GET in una riga e la routine di gestione degli errori termina con RESUME e non con GOTO. Nei programmi che usano le PRINT (o se è attiva la TRACE), il 43-esimo errore non dovuto a istruzioni INPUT causa un salto al Monitor. In tal caso se la routine di gestione degli errori termina con GOTO (invece di RESUME), l'87-esimo errore di INPUT causa un salto al Monitor.

È possibile però evitare tutti questi problemi se la vostra routine di gestione degli errori contiene una chiamata al programma in linguaggio macchina riportato nella Tabella 8.4.

Mediante POKE caricate i valori decimali nelle posizioni di memoria da 768 a 777 (o in un'altra zona disponibile). Ponete quindi CALL 768 nella vostra routine di gestione degli errori.

Non è disponibile in Integer BASIC.

Non può essere usata in modo immediato.

ON - GOSUB

Permette di saltare ad una, tra tante, subroutine a seconda del valore assunto da *esprnm* in un programma in Applesoft.

Formato:

ON *esprnm* GOSUB *linea* [, *linea...*]

Il programma salta alla subroutine della prima *linea* se *esprnm* assume il valore 1, alla subroutine della seconda *linea* se *esprnm* assume il valore 2 e così avanti. RETURN rimanda, alla fine della subroutine, all'istruzione subito successiva a ON-GOSUB.

esprnm deve avere un valore tra 0 e 255 altrimenti viene dato il messaggio ?ILLEGAL QUANTITY ERROR. Se *esprnm* assume un valore eguale a zero o superiore al numero di linee indicate, il programma prosegue subito alla istruzione successiva a ON-GOSUB.

Non è disponibile in Integer BASIC. (Vedere GOSUB per l'Integer BASIC.)

Tabella 8.4 — Subroutine in linguaggio macchina per ONERR GOTO

Linguaggio macchina		**Assembler 6502**	
Decimale	**Esadecimale**	**Istruzione**	**Commento**
104	68	PLA	Pone il byte più alto dello stack nell'Accumulatore
168	A8	TAY	e lo salva nel registro indice Y
104	68	PLA	Pone il byte seguente dello stack nell'Accumulatore
166	A6	LXD $DF	Usa il puntatore ONERR
223	DF		
154	9A	TXS	come indirizzo di stack
72	48	PHA	Spinge i contenuti salvati dello stack
152	48	TYA	nello stack 'ONERR' (due byte dall'Accu-
72	48	PHA	mulatore e dal registro Y)
96	60	RTS	Ritorna all'Applesoft

ON-GOTO

Causa il salto a una delle *linee* indicate in un programma in Applesoft.

Formato:

ON *esprnm* GOTO *linea* [, *linea...*]

Il programma prosegue alla prima *linea* se *esprnm* assume il valore 1, alla seconda *linea* se *esprnm* assume il valore 2 e così via.

esprnm deve assumere un valore compreso tra 0 e 255, altrimenti viene dato il

messaggio ?ILLEGAL QUANTITY ERROR. Se *esprnm* assume il valore zero o superiore al numero di linee indicate, il programma prosegue subito alla istruzione successiva a ON-GOTO.

Non è disponibile in Integer BASIC. (Vedere GOTO per l'Integer BASIC.)

OPEN

Prepara l'accesso ad un file sequenziale o random su disco.

Formato:

OPEN *nomefile* [, l*n*] [, D*n*] [, S*n*] [, V*n*]

Un buffer di memoria di 595 byte viene riservato per le operazioni sul file di testo specificato. Mediante i comandi READ e WRITE è possibile allora leggere e scrivere sul disco. Se il file specificato non è già presente, ne viene allore creato uno nuovo. Se il file è già aperto, allora viene chiuso e poi riaperto.

Se il parametro L manca il file è sequenziale.

Se L è presente il file è ad accesso diretto. *n* rappresenta la lunghezza in byte del record e deve essere un valore intero compreso tra 1 e 32767. Se *n* manca viene imposto L1. Ogni volta che uno stesso file, ad accesso diretto, viene aperto il valore *n* deve essere lo stesso.

Se non vi è corrispondenza nel numero di volume V*n*, viene dato l'errore VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se mancano D*n* o S*n* vengono imposti gli ultimi valori usati. Se manca V*n* viene imposto V0. Se manca *n* vengono imposti i valori D0, S0 e V0.

OPEN è un comando DOS e richiede CTRL-D in una PRINT in modo differito. OPEN non può essere usato in modo immediato.

PDL

Vedere la sezione Funzioni di questo capitolo.

PEEK

Vedere la sezione Funzioni di questo capitolo.

PLOT

Visualizza un punto sullo schermo nella grafica a bassa risoluzione.

Formato:

PLOT *col, rig*

Nella grafica a bassa risoluzione PLOT disegna un punto sullo schermo con il colore determinato dalla precedente COLOR. *col* può essere compreso tra 0 e 39; 0 è la colonna di sinistra dello schermo, 39 quella di destra. *rig* può essere compreso tra 0 e 47; 0 è la riga in alto dello schermo, 47 quella più in basso. Un punto visualizzato sulle righe da 40 a 47 è posto nella finestra di testo di quattro righe sino a che il comando POKE −16302,0 non elimini la finestra di testo.

Nel modo testo, o nella finestra di testo, PLOT disegna un carattere in colore invece di un punto. Poichè un carattere occupa lo spazio di due punti grafici in verticale, esistono due coppie di coordinate che fanno apparire un carattere sullo schermo. Per porre un particolare carattere sullo schermo, dovete dare il comando PLOT per ambedue queste metà. Il carattere che appare è determinato dall'istruzione Color eseguita prima delle PLOT.

La Tabella 8.5 riporta i caratteri generati da ogni combinazione di colore nei due punti grafici. Per generare quindi un certo carattere, ricercatelo nella Tabella 8.5; leggete i colori dei due punti grafici superiore e inferiore. La metà superiore del carattere è determinata con una PLOT su una riga pari; l'altra metà inferiore è determinata con una Plot su una riga dispari. Se tracciate solo una metà del carattere, l'altra metà viene determinata in base al colore già presente nel punto grafico.

Tabella 8.5 – Caratteri generati dalle istruzioni grafiche in modo testo

	MODO VIDEO				COLORE DEL PUNTO GRAFICO SUPERIORE															
	Video inverso	Caratteri lampeggianti	Caratteri normali		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Colore del punto grafico inferiore	0	4	8	12	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	1	5	9	13	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
	2	6	10	14	[illegible]	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
	3	7	11	15	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?

Notate che ogni carattere può essere generato da quattro diversi colori nella metà inferiore. A seconda di questo colore però, il carattere si presenterà in modo diverso. Se il colore inferiore ha un codice inferiore a 4, il carattere sarà in campo inverso; se il codice è tra 4 e 7 il carattere sarà intermittente (flashing); se il codice è superiore a 7 il carattere sarà normale.

POKE

Questo comando fa memorizzare un byte in una determinata posizione di memoria.

Formato:

POKE *memadr, byte*

Nella posizione di memoria con indirizzo *memadr* viene memorizzato un valore compreso tra 0 e 255. Se *memadr* si riferisce ad un indirizzo non disponibile (per esempio se avete soli 16K di memoria) o si riferisce ad una periferica che non può ricevere, POKE non ha effetto.

Usate questo comando solo quando siete ben sicuri di quello che fate perchè potete facilmente rovinare il software presente in memoria!

POP

Cancella il ricordo dell'indirizzo di ritorno dell'ultima GOSUB eseguita.

Formato:

POP

POP opera come se cambiasse GOSUB in GOTO. Il prossimo RETURN opererà invece normalmente facendo fare il ritorno alla penultima GOSUB eseguita. Se il numero totale di POP e RETURN supera quello delle GOSUB, viene segnalato un messaggio di errore.

POSITION

Fa avanzare di un certo numero di posizioni il puntatore di un file su disco.

Formato:

POSITION *nomefile* [, *Rn*]

POSITION fa aprire automaticamente il file *nomefile* se non è già aperto (vedere anche OPEN). Il parametro R indica di quanti campi il puntatore del file si deve spostare in avanti dalla sua posizione attuale. *n* deve essere un numero intero compreso tra 0 e 32767. Se *n* manca viene imposto R0 cioè il puntatore del file non viene mosso. Se il file viene aperto con POSITION i campi sono contati dall'inizio del file. Il file deve essere di tipo sequenziale.

Un campo è un insieme di caratteri separati dal ritorno del carrello. POSITION non fa altro che leggere carattere per carattere contando i ritorni del carrello. Se prima di arrivare al campo voluto, POSITION incontra una porzione di file vuota dà il messaggio END OF DATA.

POSITION non può essere usato in modo immediato.

POSITION è un comando DOS e richiede CTRL-D in una PRINT in modo differito.

PR#

Seleziona lo slot di una nuova periferica che riceverà i dati in uscita.

Formato:

PR# *slot*

Le PRINT successive a PR# invieranno i dati alla periferica collegata allo *slot* indicato. *slot* deve essere un valore intero compreso tra 0 e 7. Pr# 0 specifica lo schermo standard a 40 colonne. Se non vi è alcuna interfaccia inserita nello *slot* indicato, il calcolatore rimane bloccato sino a che non premete RESET.

Se il DOS è presente nella memoria del calcolatore, PR# viene considerato come un comando DOS e richiede CTRL-D in una PRINT in modo differito.

PRINT

È l'istruzione di uscita dei dati sullo schermo o su un'altra periferica di uscita.

Formato:

PRINT [espr] [$\left\{\begin{matrix};\\,\end{matrix}\right\}$... [espr]] ...]

Questa istruzione accetta molte variazioni. PRINT fa eseguire automaticamente un ritorno del carrello e un avanzamento di linea. I valori che sono associati alle espressioni listate nella PRINT vengono visualizzati. Il modo in cui tali valori appaiono dipende dalla loro natura e dalla presenza delle virgole e dei punti e virgola.

Tutti i valori numerici in Integer BASIC, e molti di loro in Applesoft, sono visualizzati nella forma numerica standard. I valori negativi sono preceduti da un segno meno; i valori positivi non sono preceduti nè dal segno più nè dallo spazio bianco. La notazione scientifica è usata in Applesoft per i valori più vicini allo zero di ±. 01 e per i valori con più di nove cifre davanti al punto decimale.

Le stringhe sono visualizzate così come sono.

Le virgole e i punti e virgola determinano la spaziatura tra i valori visualizzati. I punti e virgola fanno visualizzare i valori successivi subito dopo il valore precedente. Le virgole li fanno invece visualizzare al successivo punto di tabulazione.

In Integer BASIC i punti di tabulazione sono ogni otto caratteri alle colonne 1, 9, 17, ecc. Se un carattere, non spazio, viene visualizzato nella posizione precedente un punto di tabulazione, tale punto di tabulazione viene disattivato (per esempio la colonna 16 disattiva il punto 17).

L'Applesoft pone i punti di tabulazione ogni 16 caratteri alle colonne 1, 17, 33, ecc. Tali punti di tabulazione vengono disattivati da caratteri in posizioni precedenti come illustrato nella Figura 4.1 del capitolo 4. Per altre periferiche un punto di tabulazione viene disattivato se viene stampato un carattere, non spazio, prima del punto stesso (per esempio in colonna 32).

Se la lista delle espressioni termina senza virgola o punto e virgola, viene imposto automaticamente un ritorno carrello e un avanzamento di linea. Se la lista termina con un punto e virgola, allora una successiva PRINT farà visualizzare il primo carattere subito dopo l'ultimo carattere della PRINT attuale senza lasciare alcun spazio. Analogamente se la lista termina con una virgola la successiva PRINT inizierà a visualizzare dal prossimo punto di tabulazione.

In Integer BASIC le espressioni devono essere separate da virgole o punti e virgola. In Applesoft i caratteri separati possono essere tralasciati, ma in tal caso i valori visualizzati sarebbero concatenati come se vi fossero presenti i punti e virgola.

In Applesoft si può sostituire PRINT con la forma abbreviata (?). La parola PRINT viene in ogni caso data per intero nei listati.

READ (Istruzione DOS)

Specifica un file su disco da cui si potranno leggere i dati con INPUT e GET.

Formato:

READ *nomefile* [, R*n*] [, R*n*]

Se il file specificato non è già aperto viene aperto automaticamente (vedere anche OPEN). Tutte le istruzioni INPUT e GET successive riceveranno i loro valori dal disco sino a quando non sarà dato un altro comando DOS (oppure un CTRL-D, cioè ASCII 4, da solo), oppure sarà premuto RESET, o non si sia verificato un errore. Se il file *nomefile* non è presente sul disco, viene dato l'errore FILE NOT FOUND.

Se manca il parametro R il file sarà letto in forma sequenziale. In un file sequenziale B specifica da quale byte (carattere) iniziare la lettura del file; se manca la lettura del file inizia dal primo byte (byte 0).

Se si tenterà la lettura di un carattere, non registrato in precedenza da WRITE, apparirà il messaggio END OF DATA!

Se il parametro R è presente, il file è ad accesso diretto e R indica da quale record leggere. In tal caso il parametro B indica da quale byte, del record R, iniziare la lettura. Se B manca, la lettura inizia dal primo byte del record (byte 0). Anche in questo caso se mancano i caratteri da leggere viene dato il messaggio END OF DATA quando si eseguiranno INPUT o GET.

I valori che seguono R e B devono essere costanti intere comprese tra 0 e 32767. Se non specificati vengono imposti valori zero.

In Applesoft non potete usare CTRL-C per fermare l'istruzione READ perchè in tal caso vi apparirebbero molti messaggi ?REENTER. Usate invece RESET per fermare il programma.

READ è un comando DOS e richiede CTRL-D in una PRINT in modo differito.
Non può essere usato in modo immediato.

READ (ISTRUZIONE DI ASSEGNAZIONE)

È un'istruzione Applesoft che assegna valori tramite una DATA.

Formato:

READ *var* [, *var* ...]

Nella DATA è presente un puntatore che determina quale valore assegnare alla prima, e alle successive, variabili *var*. All'inizio dell'esecuzione del programma, o dopo una RESTORE, il puntatore punta il primo valore della DATA. Con la READ i valori della DATA vengono assegnati uno per uno alle variabili *var*.

Tra i valori della DATA e le variabili della READ vi deve essere corrispondenza di tipo. Un valore numerico può essere però attribuito ad una stringa, ma non viceversa. In questo secondo caso apparirebbe il messaggio ?SYNTAX ERROR con il numero di linea della DATA errata.

Se si tenta di assegnare più valori di quelli disponibili nella DATA, viene dato il messaggio ?OUT OF DATA ERROR con il numero di linea della READ errata.

READ può essere eseguita in modo immediato sin quando in memoria vi sia una DATA con valori sufficienti. Diversamente apparirebbe il messaggio ?OUT OF DATA ERROR.

Se è presente il DOS, READ in modo immediato viene interpretata come un comando DOS per cui appare il messaggio NOT DIRECT COMMAND.

Non è disponibile in INTEGER BASIC.

RECALL

Ricerca una variabile numerica con indici (array) su una cassetta.

Formato:

RECALL *varnm*

L'Applesoft attende di trovare la variabile sulla cassetta e nel frattempo non si possono eseguire altre istruzioni. RECALL non avvisa l'operatore di quando porre in PLAY il registratore, produce solamente un "beep" all'inizio della lettura e un altro alla fine. È necessario quindi mettere delle istruzioni di avviso PRINT prima di ogni RECALL.

La variabile con indici deve essere dimensionata prima di RECALL, altrimenti si causa l'errore ?OUT OF DATA ERROR (vedere anche DIM).

Non è necessario usare lo stesso nome per la variabile in RECALL come si era usato in STORE per memorizzarla. È importante invece che usiate lo stesso dimensionamento. Se la variabile memorizza contiene più elementi di quella richiamata, vi appare il messaggio ?OUT OF DATA ERROR. Se le due variabili contengono lo stesso

numero di elementi, ma non hanno avuto lo stesso dimensionamento, appare il messaggio ERR, ma il programma continua la sua esecuzione.

Se la variabile richiamata ha più elementi di quella registrata, i valori nella variabile richiamata possono essere inutilizzabili. Ci sono però due eccezioni. Se le due variabili hanno più indici, ma in eguale numero, allora i primi indici devono avere lo stesso dimensionamento mentre l'ultimo può essere maggiore nella variabile di RECALL. È possibile anche che la variabile di RECALL abbia più indici di quella di STORE, ma quelle in comune devono avere lo stesso dimensionamento.

Non si possono usare variabili di stringa. È possibile però convertirle in precedenza in forma numerica con le funzioni CHR$ e ASC.

Non è disponibile in Integer BASIC.

REM

Permette l'inserimento di commenti in un programma.

Formato:

REM *commento*

commento può essere una qualunque sequenza di caratteri purchè entrino in una linea.

REM compare sempre nei listati del programma, ma non viene mai tradotta in linguaggio macchina eseguibile. REM può avere una sua linea di programma o essere l'ultima istruzione su una linea con più istruzioni. REM non può assolutamente precedere un'altra istruzione, su una linea con più istruzioni, poichè tutto ciò che la segue è considerato un commento.

RENAME

Permette di cambiare il nome di un file su disco senza cambiarne il contenuto.

Formato:

RENAME $nomefile_1$, $nomefile_2$ [, D*n*] [, S*n*] [, V*n*]

Il file $nomefile_1$ cambia il nome in $nomefile_2$. Se il file era aperto viene chiuso (vedere CLOSE). Il file non subisce altre modifiche.

Attenzione: è importante che non diate al file un nome già presente su quel disco.

Se il file $nomefile_1$ non esiste sul disco nel drive D*n* dello slot S*n*, appare il messaggio FILE NOT FOUND. Se non vi è corrispondenza nel numero di volume V*n*, appare il messaggio VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se mancano D*n* o S*n* vengono imposti gli ultimi valori di drive e slot usati. V0 sostituisce V*n* se manca. Se manca *n* si usano i valori D0, S0 e V0.

RENAME è un comando DOS e richiede CTRL-D in una PRINT in modo differito.

RESTORE

Ripristina il puntatore della lista DATA. È valido solo in Applesoft.

Formato:

RESTORE

Le successive READ preleveranno i valori dall'inizio della lista della DATA.

Non è disponibile in Integer BASIC.

RESUME

Permette di riprendere l'esecuzione di un programma dall'istruzione che ha causato un errore.

Formato:

RESUME

RESUME può essere usata solo dopo che si sia verificato un errore e sia presente una ONERR GOTO. Se RESUME viene eseguita senza che sia stato commesso un errore, si può incorrere in una situazione assolutamente errata.

Non è disponibile in Integer BASIC.

Non può essere usata in modo immediato.

RETURN

Fa proseguire l'esecuzione del programma all'istruzione immediatamente successiva all'ultima GOSUB.

Formato:

RETURN

Se viene usata l'istruzione POP si perde la conoscenza dell'ultimo punto di ritorno. In tal caso RETURN fa ritornare al punto di rientro della penultima GOSUB.

Se in un programma vi sono più RETURN (e POP) che GOSUB viene dato un messaggio di errore.

ROT =

Stabilisce la rotazione delle forme grafiche, in alta risoluzione, disegnate con DRAW e XDRAW.

Formato:

ROT = *esprnm*

ROT = 0 traccia la forma con lo stesso orientamento con cui era stata definita. La forma viene ruotata di 90° in senso orario per ogni incremento di 16 nel valore di *esprnm*. Così se ROT = 32 disegna la forma all'ingiù e ROT = 64 nuovamente nella posizione originale. I valori di *esprnm* superiori a 64 sono calcolati modulo 64.

Se SCALE è stato posto eguale a 1, vi sono solo quattro possibili valori per ROT =: 0, 16, 32 e 48 (i valori superiori a 63 sono riportati a questi valori). Se SCALE è eguale a 2 ci sono invece 8 valori per ROT =, se SCALE è eguale a 3 c'è ne sono 16 e così via. Al massimo vi possono essere 64 valori per ROT =. Un valore intermedio viene riportato al valore compatibile subito inferiore.

Se *esprnm* assume un valore fuori dell'intervallo tra 0 e 255, viene dato l'errore ?ILLEGAL QUANTITY ERROR.

ROT = viene riconosciuta come parola riservata solo se è presente il segno "=".

Non è disponibile in Integer BASIC.

RUN (Istruzione DOS)

Carica ed esegue un programma da un disco.

Formato:

RUN *nomefile* [, D*n*] [, S*n*] [, V*n*]

Il programma *nomefile* viene caricato dal disco e posto in esecuzione. Ogni programma precedentemente in memoria viene cancellato.

Se il programma da caricare è in Integer BASIC e il calcolatore è in Applesoft, o viceversa, il calcolatore cambia nel linguaggio richiesto. Se necessario l'Applesoft viene caricato da disco. Se il linguaggio richiesto non è disponibile appare il messaggio LANGUAGE NOT AVAILABLE.

Se il file non esiste sul disco nel drive D*n* dello slot S*n*, viene visualizzato il messaggio FILE NOT FOUND. Se non vi è corrispondenza nel parametro V*n* viene dato l'errore VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* e S*n* mancano vengono imposti gli ultimi valori usati. VO sostituisce V*n* se manca. Se *n* manca vengono imposti i valori DO, SO e VO.

RUN è un comando DOS e richiede CTRL-D in una PRINT in modo differito.

RUN (Istruzione generale)

Pone in esecuzione il programma attualmente in memoria.

Formato generale:

RUN [*linea*]

Se presente il parametro *linea,* fa iniziare l'esecuzione proprio da questa linea. Se la linea manca viene segnalato l'errore * * * BAD BRANCH ERROR in Integer BASIC e ?UNDEF'D STATEMENT ERROR in Applesoft.

Formato aggiuntivo per l'Integer BASIC:

RUN *esprnm*

In Integer BASIC si può usare una espressione dopo RUN.
RUN può essere usato solo in modo immediato in Integer BASIC.

SAVE

Salva il programma, attualmente in memoria, su cassetta o su disco.

Formato per le cassette:

SAVE

Salva il programma, attualmente presente in memoria, sulla cassetta. Al momento della esecuzione di SAVE, il registratore deve essere in RECORD. L'Apple II non da alcun avviso per questo. Gli unici avvisi sono due "beep" all'inizio e alla fine della registrazione. Dopo il secondo "beep" potete fermare il registratore.
SAVE in Integer BASIC può essere usato solo in modo immediato.

Formato per i dischi:

SAVE *nomefile* [, D*n*] [, S*n*] [, V*n*]

Se non vi è alcun file sul disco con lo stesso nome, viene creato il nuovo file *nomefile* nel linguaggio del programma presente in memoria. Se invece esiste già un file con lo stesso nome e nello stesso linguaggio, viene allora sostituito il suo contenuto con il nuovo programma. Se infine il file esiste già, ma in un diverso linguaggio, o è un diverso tipo di file, viene dato l'errore FILE TYPE MISMATCH.
Se non vi è corrispondenza nel numero di volume V*n*, viene dato il messaggio VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* e S*n* mancano vengono imposti gli ultimi valori usati. V0 sostituisce V*n* se manca. Se manca *n* vengono imposti i valori D0, S0 e V0.

SAVE è un comando DOS e richiede CTRL-D in una PRINT in modo differito.

SCALE

Definisce l'ampiezza di scala delle forme grafiche in alta risoluzione in Applesoft.

Formato:

SCALE = *esprnm*

L'ampiezza della forma, nella tavola delle forme, è moltiplicata per il valore intero *esprnm*. Se SCALE = 1 la forma viene tracciata così come era stata disegnata; se SCALE = 2 viene tracciata con ampiezza doppia; ecc. Attenzione però che se SCALE = 0 la forma viene tracciata 255 volte più grande!

Il valore di *esprnm* deve essere compreso tra 0 e 255, altrimenti viene dato l'errore ?ILLEGAL QUANTITY ERROR.

SCALE viene riconosciuta come parola riservata solo se è seguita dal segno di eguale "=".

Non è disponibile in Integer BASIC.

SHLOAD

Questa istruzione Applesoft carica una tavola, delle forme grafiche in alta risoluzione, da una cassetta.

Formato:

SHLOAD

Nel capitolo 6 abbiamo illustrato come costruire e memorizzare una tavola delle forme (o delle figure). La tavola viene caricata subito sotto HIMEM: e poi HIMEM: viene portato subito sotto di essa. L'indirizzo iniziale della tavola è memorizzato nelle posizioni 22 e 23.

Prima di eseguire SHLOAD controllate di avere posto HIMEM: nella posizione corretta così che la tavola non si sovrapponga al vostro programma o alle sue variabili. Per maggiori chiarimenti vi rimandiamo al capitolo 6 e all'Appendice G.

Non è disponibile in Integer BASIC.

SPEED

Cambia la velocità di uscita dei caratteri.

Formato:

SPEED *esprnm*

esprnm stabilisce la velocità con cui i dati sono inviati allo schermo o su altre periferiche. I valori vanno da 0 (minima velocità) a 255 (massima velocità).
Non è disponibile in Integer BASIC.

STOP

Fa fermare l'esecuzione di un programma in Applesoft.

Formato:

STOP

Il calcolatore Apple II ritorna in modo immediato e appare il messaggio BREAK IN *linea*. *linea* è appunto il numero della linea in cui è presente STOP.
Non è disponibile in Integer BASIC.

STORE

Memorizza su cassetta una variabile con indici.

Formato:

STORE *varnm*

STORE non avvisa l'operatore quando azionare il registratore (RECORD). Dovete quindi preparare manualmente il registratore prima dell'esecuzione di STORE. È consigliabile inserire nel programma delle PRINT per avvisare l'operatore. Come per altri analoghi comandi il calcolatore suona un "beep" all'inizio e alla fine della registrazione.
Potete registrare solo variabili numeriche. Per registrare variabili di stringa con indici, usate le funzioni ASC e CHR$ (vedere anche RECALL).
Non e disponibile in Integer BASIC.

TAB

Posiziona il cursore su una determinata colonna in Integer BASIC.

Formato:

TAB *col*

Il cursore si porta sulla colonna *col* della riga corrente. Può muoversi da destra o da sinistra e non cancella i caratteri su cui passa sopra. Le colonne sono numerate da sinistra verso destra da 1 a 40.

In Applesoft dovete usare HTAB. Vedere anche la funzione TAB riportata più avanti.

TEXT

Ripristina lo schermo nel modo testo standard al posto di ogni possibile modo grafico.

Formato:

TEXT

Il carattere di pronto e il cursore vengono portati sull'ultima linea in basso dello schermo. Se il comando viene dato in modo testo si ottiene solo questo risultato.

TEXT non pulisce lo schermo cioè, più precisamente, non pulisce la pagina 1 della memoria in bassa risoluzione. Poichè il normale modo testo usa la stessa memoria della grafica in bassa risoluzione, eseguendo TEXT, durante il modo grafico in bassa risoluzione, le prime 20 linee dello schermo possono rimanere piene di caratteri spurii.

TRACE

Visualizza il numero di linea di ogni istruzione eseguita.

Formato:

TRACE

Attenzione che questo comando fa visualizzare i numeri di linea contemporaneamente alle uscite del vostro programma per cui si hanno dati forse difficilmente leggibili. Per annullare TRACE dovete dare NO TRACE.

UNLOCK

Toglie lo stato di protezione (lock) di un file.

Formato:

UNLOCK *nomefile* [, D*n*] [, S*n*] [, V*n*]

Se *nomefile* è protetto gli viene tolta la protezione; se non è protetto il comando non ha effetto. (Vedere LOCK).

Se il file non esiste sul disco nel drive D*n* dello slot S*n*, appare il messaggio FILE NOT FOUND. Se non vi è corrispondenza nel numero di volume V*n* viene dato l'errore VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* o S*n* mancano vengono imposti gli ultimi valori usati. V0 sostituisce V*n* se manca. Se manca *n* vengono imposti i valori D0, S0 e V0.

UNLOCK è un comando DOS e richiede CTRL-D in una PRINT in modo differito.

USR

Vedere la sezione Funzioni di questo capitolo.

VERIFY

Controlla la consistenza di un file su disco.

Formato:

VERIFY *nomefile* [, D*n*] [, S*n*] [, V*n*]

Quando un file viene registrato su disco con SAVE, BSAVE o PRINT viene calcolato il "checksum" per ogni settore e registrato anche lui sul disco. VERIFY ricalcola i checksum e li confronta con quelli già registrati. Se essi sono eguali il file è integro e non viene dato alcun messaggio di avviso. Se invece anche uno solo dei checksum non coincide viene segnalato l'errore I/O ERROR. VERIFY può essere usato con ogni tipo di file.

Se il file non è presente sul disco nel drive D*n* dello slot S*n*, appare il messaggio FILE NOT FOUND. Se non vi è corrispondenza nel numero di volume V*n* viene dato l'errore VOLUME MISMATCH.

D*n*, S*n* e V*n* possono essere dati con qualunque ordine. Se D*n* o S*n* mancano vengono imposti gli ultimi valori usati. V0 sostituisce V*n* se manca. Se manca *n* vengono imposti i valori D0, S0 e V0.

VERIFY è un comando DOS e richiede CTRL-D in una PRINT in modo differito.

VLIN

Traccia una linea verticale in bassa risoluzione sullo schermo.

Formato:

VLIN rig_1, rig_2 AT *col*

La linea viene tracciata sulla colonna *col* dalla riga rig_1 alla rig_2. Il colore è determinato dall'ultima istruzione COLOR eseguita. Se lo schermo è in modo testo, o è presente la finestra di testo e i valori di riga sono maggiori di 39, alcune o tutte le linee appariranno sullo schermo come caratteri e non come punti grafici. I caratteri saranno determinati in funzione dei colori scelti come abbiamo già riportato alla voce PLOT di questo capitolo (vedere anche la Tabella 8.5).

In Integer BASIC rig_1 deve essere minore o eguale di rig_2, altrimenti appare il messaggio * * * RANGE ERR.

VTAB

Posiziona il cursore sulla linea indicata e sulla colonna attuale.

Formato:

VTAB *rig*

Il cursore può muoversi in alto o in basso, per portarsi sulla linea indicata, senza cancellare i caratteri su cui passa sopra. Le righe sono numerate da 1 a 24.

WAIT

L'esecuzione di un programma si ferma sino a che una certa posizione di memoria assume un determinato valore.

Formato:

WAIT *memadr* , $esprnm_1$ [, $esprnm_2$]

WAIT controlla i bit contenuti all'indirizzo di memoria *memadr* se sono eguali ai bit indicati da $esprnm_2$ in funzione, però, di una maschera determinata da $esprnm_1$. La maschera opera nel modo seguente: degli otto bit di $esprnm_1$ vengono presi in considerazione solo quelli che hanno valore 1; le stesse posizioni di questi bit sono quelle che sono controllate poi tra $esprnm_2$ e l'indirizzo *memadr.*

Sino a che tutti i bit tra $esprnm_2$ e il contenuto di *memadr* non saranno eguali, secondo lo schema di $esprnm_1$, il calcolatore non procederà alla istruzione successiva.

Se $esprnm_2$ manca viene imposto il valore 0.

WAIT può essere fermata solo da RESET o spegnendo il calcolatore. I valori di $esprnm_1$ e $esprnm_2$ devono essere compresi ra 0 e 255 altrimenti viene dato l'errore ?ILLEGAL QUANTITY ERROR. Se *memadr* si riferisce ad un indirizzo non disponibile (per esempio 35436 se avete solo 32K di memoria) o ad un indirizzo di periferica non presente, il calcolatore si blocca sino a che non premiate RESET.

Non è disponibile in Integer BASIC.

WRITE

Specifica un file su disco a cui saranno inviati con PRINT dei dati in scrittura.

Formato:

WRITE *nomefile* [, R*n*] [, B*n*]

Se il file non è già aperto, viene aperto automaticamente (vedere OPEN). Le successive istruzioni PRINT invieranno i loro dati sul disco sino a quando non sarà data un'altra istruzione per il disco oppure sarà dato CTRL-D (ASCII 4). Se il file non è sul disco appare il messaggio FILE NOT FOUND.

Se manca il parametro R il file viene trattato in forma sequenziale. In tal caso il parametro B indica da quale carattere (byte) inizierà la scrittura. Se B manca la scrittura inizierà dal primo byte (byte 0).

Se invece R è presente il file è ad accesso diretto. WRITE scriverà dal record R*n*. In tal caso B indica il byte all'interno del record R da cui inizierà la scrittura. Se in quest'ultimo caso B manca la scrittura inizierà dal primo byte (byte 0) del record R.

Il parametro B può essere usato per scrivere oltre l'ultimo carattere già presente nel file. Ricordatevi però che se tentate di leggere oltre l'ultimo carattere vi viene segnalato l'errore OUT OF DATA.

I valori che seguono R e B devono essere interi compresi tra 0 e 32767. Se mancano sono posti eguali a 0.

Mentre WRITE è attiva, tutti i dati che il calcolatore invierebbe allo schermo, compreso il punto interrogativo delle INPUT, vanno invece sul disco. Sul disco vanno anche i messaggi di errore!

WRITE è un comando DOS e richiede CTRL-D in una PRINT in modo differito. Non può essere usato in modo immediato.

XDRAW

Questa istruzione Applesoft traccia una forma grafica in alta risoluzione sullo schermo. Se eseguita una seconda volta con gli stessi parametri cancella invece la figura.

Formato:

XDRAW *esprnm* [AT *colh* , *righ*]

La forma numero *esprnm* della tavola delle forme viene tracciata con i colori punto per punto complementari a quelli già presenti sullo schermo. I colori 0 e 3 sono complementari, come anche 1 e 2, 4 e 7, 5 e 6 (vedere la Tabella 8.3). La scala e la rotazione della figura devono essere stabiliti con SCALE e ROT prima di XDRAW.

XDRAW viene usata al posto di DRAW per poter facilmente cancellare una figura già tracciata. Siccome XDRAW disegna nel colore complementare a quello presente sullo schermo, se la eseguite un numero pari di volte con gli stessi parametri il suo disegno non rimane sullo schermo.

Se in XDRAW non viene indicato il punto di partenza della figura, il disegno inizierà dall'ultimo punto tracciato da DRAW, XDRAW o HPLOT. Altrimenti la figura inizia dal punto *colh, righ.*

esprnm deve avere un valore tra 0 e il numero massimo di forme presenti nella tavola delle figure (0 - 255).

Non è disponibile in Integer BASIC.

FUNZIONI

Le funzioni BASIC dell'Apple II sono riportate qui di seguito in ordine alfabetico. La nomenclatura è la stessa usata per le istruzioni.

Molte funzioni sono disponibili solo in Applesoft.

ABS

Ritorna il valore assoluto di un numero. Cioè il valore del numero prescindendo dal segno.

Formato:

ABS *esprnm*

ASC

Ritorna il codice numerico ASCII di un carattere.

Formato:

ASC (*espr*$)

Se la stringa è più lunga di un carattere, la funzione ritorna il codice solo del primo carattere. Il codice ritornato non è necessariamente il valore più basso per quel carattere e compreso quindi tra 0 e 95, ma può essere anche un valore tra 96 e 255. Attenzione però che questa coppia di caratteri non è trattata come lo stesso carattere dagli operatori <, > e =. Alcune periferiche di uscita possono anche trattarli diversamente. Se il carattere è CTRL-@ (codice ASCII 0) viene dato il messaggio ?SYNTAX ERROR. Se *espr*$ è una stringa nulla viene dato il messaggio ?ILLEGAL QUANTITY ERROR.

I codici ASCII sono riportati nell'Appendice I.

ATN

Ritorna l'arcotangente dell'argomento.

Formato:

ATN (*esprnm*)

Il risultato viene dato in radianti tra - $\pi/2$ e $\pi/2$.

Non è disponibile in Integer BASIC.

CHR$

Ritorna il valore di stringa di un determinato codice numerico ASCII.

Formato:

CHR$ (*esprnm*)

Viene dato il carattere corrispondente secondo il codice ASCII. Nell'appendice I sono riportati i codici ASCII. È importante usare questa funzione per generare quei caratteri che, se battuti alla tastiera, sono subito esecutivi come per esempio il tasto RETURN. *esprnm* deve avere un valore compreso tra 0 e 255, altrimenti appare il messaggio ?ILLEGAL QUANTITY ERROR.

Non è disponibile in Integer BASIC.

COS

Ritorna il coseno di un angolo.

Formato:

COS (*esprnm*)

L'argomento deve essere dato in radianti.

Non è disponibile in Integer BASIC.

EXP

Ritorna *e* elevato alla potenza indicata.

Formato:

EXP (*esprnm*)

Calcola *e* elevato a *esprnm* (*e* è la base dei logaritmi naturali 2.71828183).

Non è disponibile in Integer BASIC.

FN

Richiama un funzione di utente già in precedenza definita.

Formato:

FN *varnm* (*esprnm*)

varnm è il nome della funzione. Il valore di *esprnm* viene assegnato alla variabile fittizia, presente nella definizione della funzione, e poi viene calcolata l'espressione risultante. Per ulteriori chiarimenti vedere DEF FN.

Una funzione non può essere ricorsiva cioè in *esprnm* non può comparire la funzione stessa, FN *varnm*, nè un'altra funzione che faccia poi riferimento alla FN *varnm*.

FN *varnm* deve essere usata dopo DEF FN *varnm*, altrimenti vi viene segnalato l'errore ?UNDEF' D FUNCTION ERROR.

Non è disponibile in Integer BASIC.

FRE

Ritorna il numero di byte di memoria disponibili per un programma in Applesoft.

Formato:

FRE (*esprnm*)

La memoria disponibile è quella compresa tra l'area di memorizzazione delle stringhe e quella di memorizzazione delle array. Se vi sono più di 32767 byte liberi, FRE ritorna un numero negativo a cui dovete sommare 65536 per ottenere l'ammontare effettivo di memoria libera.

FRE svolge anche il compito di togliere, dall'area di memorizzazione delle stringhe, le stringhe non più usate. Quando una stringa cambia valore durante l'esecuzione di un programma, il suo vecchio valore rimane in memoria e quello nuovo viene aggiunto nell'area di memorizzazione delle stringhe. Ovviamente ciò comporta un cattivo uso della memoria. Per ovviare a questo basta usare più volte FRE, in un programma, per ricuperare la memoria occupata dalle vecchie stringhe. Per esempio ponete A = FRE (0) in vari punti del vostro programma.

L'argomento *esprnm* di FRE è solo fittizio, ma se non viene citato si causa un errore.

Non è disponibile in Integer BASIC.

INT

Ritorna la parte intera di un numero.

Formato:

INT (*esprnm*)

Ritorna il numero intero immediatamente inferiore o eguale a *esprnm*.

Non è disponibile in Integer BASIC.

LEFT$

Ritorna i caratteri più a sinistra della stringa.

Formato:

LEFT$ (*espr$* , *esprnm*)

Ritorna *esprnm* caratteri più a sinistra della stringa *espr$*. *esprnm* deve essere compreso tra 1 e 255 mentre *espr$* non può avere più di 255 caratteri. Se *esprnm* è maggiore della lunghezza della stringa *espr$*, allora viene ritornata tutta la stringa.

Non è disponibile in Integer BASIC.

LEN

Ritorna la lunghezza di una stringa.

Formato:

LEN (*espr$*)

LEN conta tutti i caratteri della stringa *espr$* compresi gli spazi e i caratteri non stampabili. Se la lunghezza è maggiore di 255 caratteri (ciò è possibile solo se *espr$* è un'espressione ottenuta con il concatenamento di più stringhe) allora viene dato il messaggio ?STRING TOO LONG ERROR.

LOG

Ritorna il logaritmo naturale dell'argomento.

Formato:

LOG (*esprnm*)

Calcola il logaritmo naturale di *esprnm.* Se *esprnm* è zero o negativo viene dato l'errore ?ILLEGAL QUANTITY ERROR.

Non è disponibile in Integer BASIC.

MID$

Ritorna una parte interna di una stringa.

Formato:

MID$ (*espr$* , $esprnm_1$, $esprnm_2$)

A partire dal carattere $esprnm_1$ ritorna $esprnm_2$ caratteri verso destra della stringa *espr*$. Se manca $esprnm_2$, MID$ ritorna tutti i caratteri di destra da $esprnm_1$. Se la stringa è più corta di $esprnm_1$, viene ritornata la stringa nulla. Se dopo $esprnm_1$ vi sono nella stringa meno caratteri di $esprnm_2$, ciò equivale all'assenza di $esprnm_2$. La stringa non può avere più di 255 caratteri; $esprnm_1$ e $esprnm_2$ devono essere compresi nell'intervallo tra 1 e 255.

Non è disponibile in Integer BASIC.

PDL

Ritorna il valore corrente del controllore dei giuochi (paddle) specificato.

Formato:

PDL (*esprnm*)

Il valore ritornato è un intero compreso tra 0 e 255 basato sulla rotazione della manopola del paddle numero *esprnm,* o basato sulla resistenza di un dispositivo connesso al connettore dei giuochi *esprnm.* I controllori per giuochi possono essere numerati da 0 a 3. Se il numero *esprnm* è minore di zero, o maggiore di 255, viene visualizzato il messaggio ?ILLEGAL QUANTITY ERROR; se è compreso tra 4 e 255, la funzione PDL ritorna un valore assolutamente imprevedibile compreso tra 0 e 255 e può inoltre causare un click nell'altoparlante o far passare il calcolatore al modo grafico, o altro ancora.

Se fate eseguire due funzioni PDL troppo vicine, rischiate che la seconda sia influenzata dalla prima. Cercate di separare le due PDL da un conveniente numero di altre istruzioni nei vostri programmi (per esempio eseguite un ciclo FOR-NEXT di attesa).

PEEK

Ritorna il contenuto di una posizione di memoria.

Formato:

PEEK (*memadr*)

Il valore ritornato è l'equivalente decimale degli otto bit contenuti nella posizione di memoria *memadr.* In Appendice E sono riportate alcune importanti posizioni di memoria.

POS

Ritorna il numero di colonna su cui è presente il cursore.

Formato:

POS (*esprnm*)

L'argomento *esprnm* è fittizio e può quindi assumere un valore qualunque.
POS ritorna un valore compreso tra 0 e 39.
Non è disponibile in Integer BASIC.

RIGHT$

Ritorna la parte destra di una stringa.

Formato:

RIGHT$ (*espr$* , *esprnm*)

Ritorna gli *esprnm* caratteri di destra della stringa *espr$*. *esprnm* deve essere compreso tra 1 e 255 mentre *espr$* non può avere più di 255 caratteri. Se *esprnm* è maggiore della lunghezza di *espr$*, viene ritornata tutta la stringa.
Non è disponibile in Integer BASIC.

RND

Genera dei numeri a caso (random).

Formato:

RND (*esprnm*)

Ritorna un valore a caso compreso in un intervallo che dipende dall'argomento *esprnm* e dal tipo di BASIC.

In Integer BASIC, RND ritorna un valore intero compreso tra 0 (incluso) e *esprnm* (escluso). Ne consegue che RND (1) è sempre 0, mentre RND (–2) produce delle alternanze di 0 e –1. Se date RND (0) commettete un errore che vi fa apparire il messaggio * * * > 32767 ERR.

In Applesoft invece RND genera numeri reali maggiori o eguali a 0 e minori di 1. La sequenza generata può essere di tre tipi a seconda del segno di *esprnm*.

Se *esprnm* è positivo, RND ritorna un valore diverso ogni volta che viene usato, a meno che sia stata iniziata una sequenza ripetibile.

Una sequenza ripetibile inizia quando *esprnm* è negativo. Ogni particolare valore negativo fa iniziare la stessa sequenza; valori positivi successivi fanno poi ritornare una sequenza ripetibile di numeri a caso. Sequenze diverse di numeri a caso sono inizializzate ciascuna con valori negativi di *esprnm*. Questa caratteristica è utile per provare programmi che usano i numeri a caso.

Se *esprnm* è eguale a 0 in Applesoft, RND ritorna il valore a caso generato per ultimo (ciò non è influenzato da CLEAR e NEW).

SCRN

Ritorna il codice di colore, del punto posto alle coordinate indicate, nella grafica in bassa risoluzione.

Formato:

SCRN (*col* , *rig*)

Se lo schermo è in modo testo, o è presente la finestra di testo e il punto è contenuto in essa, SCRN ritorna il codice di colore di mezzo carattere. Viene ritornato il codice di colore della metà superiore del carattere se *rig* è pari; quello della metà inferiore se *rig* è dispari. Il codice ASCII del carattere nella posizione (*a*,*b*) (con *a* compreso tra 0 e 39 e *b* compreso tra 0 e 23) viene ritornato dall'espressione SCRN (*a*, 2 * *b*) + 16 * SCRN (*a*, 2 * *b* + 1). Il carattere può poi essere dato da CHR$ (*n*) dove *n* è il risultato dell'espressione precedente.

Se *col* è compreso tra 0 e 39, SCRN ritorna quindi il colore del punto alle coordinate (*col* , *rig*). Se *col* è compreso tra 40 e 47 e *rig* è compreso tra 0 e 31, SCRN ritorna il colore del punto grafico (*col* – 40, *rig* + 16). Se *col* è compreso tra 40 e 47 e *rig* tra 32 e 47, SCRN ritorna un numero assolutamente aleatorio.

Se SCRN viene usato, mentre lo schermo è in modo grafico ad alta risoluzione, il valore ritornato dipende dalla memoria grafica in bassa risoluzione e non dall'immagine in alta risoluzione.

SCRN viene riconosciuta come parola riservata solo se il primo carattere successivo, non spazio, è una parentesi sinistra.

SGN

Determina se un numero è positivo, nullo o negativo.

Formato:

SGN (*esprnm*)

SGN ritorna il valore +1 se *esprnm* è positivo, –1 se è negativo e 0 se è nullo.

SIN

Ritorna il seno di un angolo.

Formato:

SIN (*esprnm*)

Calcola il valore seno dell'angolo *esprnm* espresso in radianti.
Non è disponibile in Integer BASIC.

SPC

Muove il cursore verso destra di un determinato numero di posti.

Formato:

SPC (*esprnm*)

SPC viene usata in una PRINT per far visualizzare un certo numero di spazi bianchi. I caratteri su cui passa sopra il cursore vengono quindi cancellati.

SPC muove il cursore verso destra di *esprnm* colonne indipendentemente da quale posizione si trovava. Ciò è diverso da TAB che porta il cursore su posizioni predeterminate. (vedere la funzione TAB).

Non è disponibile in BASIC.

SQR

Ritorna la radice quadrata di un numero positivo.

Formato:

SQR (*esprnm*)

Se *esprnm* è negativo viene segnalato l'errore ?ILLEGAL QUANTITY ERROR. SQR (*esprnm*) viene calcolata più rapidamente di (*esprnm*) ^ (.5).

Non è disponibile in Integer BASIC.

STR$

Converte un valore numerico in stringa.

Formato:

STR$ (*esprnm*)

Il valore *esprnm* viene convertito in stringa. I caratteri che si ottengono sono gli stessi che sarebbero visualizzati con PRINT *esprnm.* Per esempio STR$ (2/3) = ".666666667" e STR$ (2468222579) = "2.46822258E + 09". Se *esprnm* supera il limite dei numeri reali, appare il messaggio ?OVERFLOW ERROR.

Non è disponibile in Integer BASIC.

TAB

Porta il cursore sulla colonna indicata.

Formato:

TAB (*esprnm*)

TAB viene inserita in una PRINT per portare il cursore sulla colonna *esprnm,* se tale colonna è a destra della posizione attuale del cursore. Il cursore non si muove se *esprnm* non è a destra del cursore. TAB visualizza degli spazi bianchi per cui cancella i caratteri su cui passa sopra.

Per TAB le colonne sono numerate da 1 a 255. Se *esprnm* è maggiore della larghezza della periferica di uscita (per esempio 40 per lo schermo televisivo), TAB prosegue alla linea inferiore e prosegue il conteggio ad iniziare dall'estremo sinistro. Se *esprnm* è eguale a zero, TAB si muove alla posizione 256. Per valori esterni a 0 - 255 appare il messaggio ?ILLEGAL QUANTITY ERROR.

Consultate anche HTAB (in Applesoft) e TAB (in Integer BASIC) nella sezione Istruzioni di questo capitolo.

Non è disponibile in Integer BASIC.

TAN

Ritorna la tangente di un angolo.

Formato:

TAN (*esprnm*)

Calcola la tangente dell'angolo *esprnm* espresso in radianti.

Non è disponibile in Integer BASIC.

USR

Fa eseguire un salto ad una subroutine in linguaggio macchina passando i parametri all'Accumulatore.

Formato:

USR *esprnm*

La subroutine inizia all'indirizzo 10 (0A in esadecimale). Gli indirizzi da 10 a 12 (0A, 0C in esadecimale) devono contenere una istruzione di salto JMP, in assembler, che fa eseguire un salto all'indirizzo iniziale della vostra subroutine. Siccome USR è una funzione, essa ritorna un valore reale. Tale valore ritornato è il contenuto dell'Ac-

cumulatore quando termina la subroutine e viene quindi eseguita l'istruzione RTS (ritorno al programma Applesoft).

Nell'Appendice D sono riportate alcune subroutine contenute nel Monitor dell'Apple II.

Consultate anche l'istruzione CALL, già presentata in questo capitolo, che è disponibile anche in Integer BASIC.

Non è disponibile in Integer BASIC.

VAL

Converte una stringa in valori numerici.

Formato:

VAL (*espr*$)

Ritorna il valore numerico rappresentato da *espr*$. Se il primo carattere di *espr*$ non è numerico viene ritornato il valore zero. Altrimenti viene preso carattere per carattere sino a che non si incontra un carattere non accettabile. I caratteri accettabili sono: le cifre da 0 a 9, gli spazi, il punto decimale, i segni più o meno e, nel caso di notazione scientifica del numero, la lettera E.

Se *espr*$ mediante concatenamento contiene più di 255 caratteri, allora appare il messaggio ?STRING TOO LONG ERROR. Se il valore numerico supera il valore massimo consentito per i reali, viene dato il messaggio ?OVERFLOW ERROR.

Non è disponibile in Integer BASIC.

APPENDICE A

ALCUNE FUNZIONI NUMERICHE

Riportiamo alcune delle più comuni funzioni numeriche. Attenzione però che alcuni valori della x possono invalidare la funzione (per esempio se COS (x) = 0 allora SEC (x) è immaginario). Nei vostri programmi dovete quindi fare un controllo dei valori assunti dalla variabile x.

Nessuna di queste funzioni è valida in Integer BASIC.

ARCCOS (x) = –ATN (x/ SQR (– * + 1)) + 1.5707633
Ritorna la funzione inversa del coseno di x (ABS (x) < 1).

ARCCOT (x) = –ATN (x) + 1.5707633
Ritorna la funzione inversa della cotangente di x.

ARCCOSH (x) = LOG (x + SQR (x * x – 1))
Ritorna la funzione inversa del coseno iperbolico di x (x > =1).

ARCCOTH (x) = LOG ((x + 1) / (x – 1)) /2
Ritorna la funzione inversa della cotangente iperbolica di x (ABS (x) > 1).

ARCCSC (x) = ATN (1/SQR (x * x – 1)) + (SGN (x) –1) * 1.5707633
Ritorna la funzione inversa della cosecante di x (ABS (x) > 1).

ARCCSCH (x) = LOG ((SGN (x) * SQR (x * x + 1) + 1) /x)
Ritorna la funzione inversa della cosecante imperbolica di x (x > 0).

ARCSEC (x) = ATN (SQR (x * x – 1)) + (SGN (x) – 1) * 1.5707633
Ritorna la funzione inversa della secante di x (ABS (x) > = 1).

ARCSECH (x) = LOG ((SQR (– x *x + 1) + 1)/x)
Ritorna la funzione inversa della secante iperbolica di x (0 < x < = 1).

ARCSIN (x) = ATN (x/SQR (– x * x + 1))
Ritorna la funzione inversa del seno di x (ABS (x) <1).

ARCSINH (x) = LOG (x + SQR (x * x + 1))
Ritorna la funzione inversa del seno iperbolico di x.

ARCTANH (x) = LOG ((1 + x) / (1 – x)) / 2
Ritorna la funzione inversa della tangente iperbolica di x (ABS (x) <1).

COSH (x) = (EXP (x) + EXP (– x)) /2
Ritorna il coseno imperbolico di x.

COT (x) = 1/TAN (x)
Ritorna la cotangente di x (x < > 0).

CSC (x) = 1/SIN (x)
Ritorna la cosecante di x (x < > 0).

CSCH (x) = 2/(EXP (x) – EXP (– x))
Ritorna la cosecante iperbolica di x (x < > 0).

LOG_a (x) = LOG (x) / LOG (a)
Ritorna il logaritmo in base a di x (a > 0, x > 0).

LOG_{10} (x) = LOG (x) / 2.30258509
Ritorna il logaritmo in base 10 di x (x > 0).

SEC (x) = 1/COS (x)
Ritorna la secante di x (x < > π /2).

SECH (x) = 2/(EXP (x) + EXP (– x))
Ritorna la secante iperbolica di x.

SINH (x) = (EXP (x) – EXP (– x))/2
Ritorna il seno iperbolico di x.

TANH (x) = (EXP (x) – EXP (– x)) / (EXP (x) + EXP (– x))
Ritorna la tangente iperbolica di x.

APPENDICE B

COMANDI DI EDITING

In questa Appendice riportiamo i comandi di Editing dati direttamente alla tastiera.

→ Muove il cursore in avanti sulla linea corrente. I caratteri su cui passa sopra il cursore sono ricopiati in memoria come se fossero battuti in quel momento. Lo schermo non viene modificato.

← Muove il cursore indietro sulla linea corrente. I caratteri su cui passa sopra il cursore sono cancellati dalla memoria, ma non sullo schermo.

REPT Fa ripetere la battitura di un carattere premuto contemporaneamente. Il tasto REPT deve essere premuto *dopo* l'altro tasto.

CTRL-X Viene annullata la linea sullo schermo corrente mentre il cursore va a capo all'inizio della linea successiva.

Sequenze ESC

I sette comandi seguenti sono ottenuti premendo prima il tasto ESC, poi rilasciando e premendo quindi il secondo tasto.

ESC - A Muove il cursore di una posizione a destra senza modificare ne lo schermo ne la memoria.

ESC - B Muove il cursore di una posizione a sinistra senza modificare nè lo schermo nè la memoria.

ESC - C Muove il cursore di una posizione in basso senza modificare nè lo schermo nè la memoria.

ESC - D Muove il cursore di una posizione in alto senza modificare nè lo schermo nè la memoria.

ESC - E Cancella tutti i caratteri dal cursore all'estremo destro della linea corrente.

ESC - F Cancella tutti i caratteri dal cursore alla fine dello schermo.

ESC - @ Pulisce lo schermo e riporta il cursore in alto a sinistra.

Comandi in modo *edit*

I quattro comandi seguenti richiedono l'Autostart Monitor e possono essere dati solo in modo *edit.* Richiamate il modo *edit* premendo il tasto ESC e cambiatelo poi premendo un qualunque tasto diverso da I, J, K, M, REPT, CTRL o SHIFT.

I Muove il cursore di una posizione in alto senza cambiare il modo edit.

J Muove il cursore di una posizione a sinistra senza cambiare il modo edit.

K Muove il cursore di una posizione a destra senza cambiare il modo edit.

M Muove il cursore di una posizione in basso senza cambiare il modo edit.

APPENDICE C

MESSAGGI DI ERRORE

I messaggi di errore sono ripartiti in tre categorie: quelli relativi all'Integer BASIC, quelli relativi all'Applesoft e quelli che riguardano il DOS.

I messaggi DOS e la maggior parte di quelli Applesoft hanno associato un codice di errore. Nel caso di una istruzione ONERR GOTO il codice di errore si può trovare all'indirizzo 222.

Nella Tabella C.1 sono riportati i messaggi di errore in ordine di numero di codice (posizione 222).

MESSAGGI DI ERRORE DELL'INTEGER BASIC

*** > 255 ERR

Un valore che dovrebbe essere tra 0 e 255 è invece fuori da questo intervallo.

*** >32767 ERR

È stato superato l'intervallo numerico tra –32767 e 32767.

*** 16 FORS ERR

Sono stati inizializzati più di 16 cicli FOR.

*** 16 GOSUBS ERR

Sono attivi più di 16 salti GOSUB.

*** BAD BRANCH ERR

È stata data una istruzione di salto ad una linea non esistente.

*** BAD NEXT ERR

È stato incontrato un NEXT senza il corrispondente FOR.

*** BAD RETURN ERR

Esistono più RETURN che corrispondenti GOSUB.

*** DIM ERR

Una stessa variabile con indici è stata dimensionata più volte.

*** MEM FULL ERR

È richiesta più memoria di quella disponibile.

*** NO END ERR

L'ultima istruzione fisica di un programma non è una END.

*** RANGE ERR

L'indice di una variabile con indici è stato imposto minore di zero o maggiore del valore massimo previsto dal dimensionamento. Oppure l'argomento di HLIN, VLIN, PLOT, TAB o VTAB è esterno all'intervallo consentito.

RETYPE LINE

Un errore è stato generato in risposta ad una INPUT. Dopo un messaggio diagnostico viene data questa direttiva.

*** STRING ERR

È stata effettuata una operazione di stringa illegale.

*** STR OVFL ERR

Ad una stringa sono stati assegnati più caratteri di quanti previsti dal suo dimensionamento.

*** SYNTAX ERR

È un errore di sintassi oppure un errore non previsto da un altro messaggio.

*** TOO LONG ERR

Si sono usati più di 12 livelli di parentesi oppure sono state scritte linee più lunghe di 128 caratteri.

MESSAGGI DI ERRORE DELL'APPLESOFT

?BAD SUBSCRIPT ERROR

Gli indici di una variabile, con indici, sono in numero sbagliato oppure superano il valore previsto dal dimensionamento. Codice di errore 107.

?CAN'T CONTINUE ERROR

Si è tentato di continuare un programma (con il comando CONT) che non può continuare; per esempio dopo un errore fatale o perchè il programma non c'è più.

?DIVISION BY ZERO ERROR

Si è tentato di eseguire una divisione con divisore nullo. Codice di errore 133.

?FORMULA TOO COMPLEX ERROR

Più di due istruzioni del tipo IF *stringa* THEN sono state eseguite. Codice di errore 191.

?ILLEGAL DIRECT ERROR

È stato dato in modo immediato uno dei comandi INPUT, DEF FN o GET.

?ILLEGAL QUANTITY ERROR

Un valore numerico è esterno all'intervallo consentito per una funzione di stringa, per una funzione numerica, per una istruzione grafica o per altri casi. Codice di errore 53.

?NEXT WITHOUT FOR ERROR

Si è incontrato un NEXT senza il corrispondente FOR. Un NEXT senza il nome della variabile di ciclo genera questo errore solo se non vi è alcun FOR attivo. Codice di errore 0.

?OUT OF DATA ERROR

Si è tentato di leggere più elementi DATA di quelli disponibili. Codice di errore 42.

?OUT OF MEMORY ERROR

Può essere dovuto ad una di queste cause: programma troppo lungo, troppe variabili, più di 10 livelli di cicli FOR, più di 24 richiami di subroutine, più di 36 livelli di parentesi, LOMEM: posto troppo in alto o HIMEM: troppo in basso. Codice di errore 77.

?OVERFLOW ERROR

Un numero è uscito dall'intervallo tra —1.7 E + 38 e 1.7 E + 38. Codice di errore 69.

?REDIM'D ARRAY ERROR

Viene ripetuto il dimensionamento di una variabile con indici (array). Spesso questo errore è dovuto al fatto che il primo dimensionamento è avvenuto per default. Codice di errore 120.

?RETURN WITHOUT GOSUB ERROR

Esistono più RETURN che GOSUB. Codice di errore 22.

?STRING TOO LONG ERROR

Si è tentato di concatenare più stringhe per un totale superiore a 255 caratteri. Codice di errore 176.

?SYNTAX ERROR

È un errore di sintassi, cioè relativo al formato delle istruzioni, oppure un errore non previsto dagli altri messaggi. Codice di errore 16.

?TYPE MISMATCH ERROR

Si è usata una grandezza numerica al posto di una stringa o viceversa. Oppure non vi è concordanza di tipo a sinistra e a destra del segno di eguale di una assegnazione. Codice di errore 163.

?UNDEF'D FUNCTION ERROR

Si è richiamata una funzione di utente che non è stata definita. Codice di errore 224.

?UNDEF'D STATEMENT ERROR

Si è tentato di fare un salto ad una istruzione che non esiste. Codice di errore 90.

MESSAGGI DI ERRORE DEL DOS

DISK FULL

Si è cercato di scrivere su un disco più dati di quanti ne può contenere. Nel caso di un disco pieno, questo messaggio può essere dato anche al posto di messaggi più opportuni (per esempio FILE NOT FOUND). Codice di errore 9.

END OF DATA

Si è tentato di leggere una parte di un file di testo che non è stata mai scritta. Codice di errore 5.

FILE LOCKED

Si è tentato di accedere ad un file protetto (locked) con SAVE, BSAVE, WRITE, DELETE o RENAME. Codice di errore 10.

FILE NOT FOUND

Si tenta di accedere ad un file che non esiste sul disco. Questo messaggio viene dato quando il DOS, dopo avere visto la mancanza del file, non ne crea uno nuovo. Codice di errore 6.

FILE TYPE MISMATCH

Si è cercato di accedere ad un file che non è del tipo richiesto. I comandi LOAD, RUN e SAVE possono essere usati solo con file di programma. Il comando CHAIN può essere usato solo con file di programma in Integer BASIC. I comandi OPEN, READ, WRITE, APPEND, POSITION e EXEC possono essere usati solo con file di testo. I comandi BLOAD, BSAVE e BRUN possono essere usati solo con file binari. Codice di errore 13.

I/O ERROR

Una operazione di ingresso/uscita su disco non ha avuto successo. Alcune cause possibili sono: il portellino del drive è aperto, il disco non è stato inizializzato, manca il disco nel drive o il disco è difettoso. Codice di errore 8.

LANGUAGE NOT AVAILABLE

Con i comandi FP o INT si è cercato di richiamare un linguaggio che non è disponibile nelle ROM o su disco. Analogamente si è cercato di eseguire un programma in un linguaggio non disponibile. Codice di errore 1.

Tabella C.1 — Codice degli errori

PEEK (222)	Tipo di errore	Linguaggio
0	NEXT senza FOR	Applesoft
1	Linguaggio non disponibile	DOS
2 o 3	Errore di intervallo	DOS
4	Protezione contro la scrittura	DOS
5	Fine dei dati	DOS
6	File non esistente	DOS
7	Incompatibilità di volume	DOS
8	Errore di I/O	DOS
9	Disco pieno	DOS
10	File protetto	DOS
11	Errore di sintassi	DOS
12	Mancanza di buffer disponibili	DOS
13	Incompatibilità di tipo di file	DOS
14	Programma troppo lungo	DOS
15	Comando non di tipo immediato	DOS
16	Errore di sintassi	Applesoft
22	RETURN senza GOSUB	Applesoft
42	Mancanza di dati in DATA	Applesoft
53	Quantità illegale	Applesoft
69	Superamento "overflow"	Applesoft
77	Superamento della memoria	Applesoft
90	Istruzione non definita	Applesoft
107	Indice errato	Applesoft
120	Ripetizione di un dimensionamento	Applesoft
133	Divisione per zero	Applesoft
163	Incompatibilità di tipo	Applesoft
176	Stringa troppo lunga	Applesoft
191	Formula troppo complessa	Applesoft
224	Funzione non definita	Applesoft
254	Errata risposta ad una INPUT	Applesoft
255	È stato premuto CTRL-C	Applesoft

NO BUFFERS AVAILABLE

Tutti i buffer disponibili sono già usati e se ne richiede uno nuovo. Codice di errore 12.

NOT DIRECT COMMAND

I seguenti comandi DOS possono essere dati solamente in modo differito con una PRINT: APPEND, OPEN, POSITION, READ e WRITE. Codice di errore 15.

PROGRAM TOO LARGE

Un comando DOS ha cercato di caricare in memoria RAM un file troppo grande per la memoria disponibile. Codice di errore 14.

RANGE ERROR

Un parametro di un comando DOS è fuori dai limiti consentiti. Per esempio il parametro di drive D può essere solo 1 o 2. Codice di errore 2 o 3.

SYNTAX ERROR

Un comando DOS ha un errore di formato. Codice di errore 11.

VOLUME MISMATCH

Non vi è concordanza tra il parametro di volume V di un comando DOS e il valore effettivo sul disco. Codice di errore 7.

WRITE PROTECTED

Si è dato un comando SAVE, BSAVE o WRITE su un disco protetto in scrittura. Codice di errore 4.

APPENDICE D

ALCUNE SUBROUTINE DI SISTEMA

Nelle tabelle seguenti sono riportate alcune subroutine di Sistema in linguaggio macchina disponibili nel calcolatore Apple II. La Tabella D.1 elenca queste subroutine a seconda della loro funzione e indica qual è il loro punto di inizio (entry point).

La Tabella D.2 riporta le stesse subroutine in ordine di punto di inizio e indica anche quale deve essere l'informazione contenuta nei registri pima di eseguire la subroutine (colonna terza). Viene anche segnalato (colonna quarta) quali registri sono interessati durante l'esecuzione della subroutine.

La maggior parte di queste subroutine possono essere chiamate direttamente con un comando BASIC oppure mediante una CALL. Se nell'ultima colonna della Tabella D.2 compare una A, ciò significa che la subroutine è disponibile solo in Applesoft.

Alcune subroutine non hanno però un equivalente in ambedue le versioni di BASIC e quindi non possono essere eseguite con una semplice CALL perchè alcuni registri devono essere caricati prima dell'esecuzione della subroutine. Per caricare i registri vi sono metodi diversi a seconda del tipo di BASIC.

In Integer BASIC la soluzione è abbastanza semplice. Eseguite dapprima una CALL -182 per salvare il contenuto attuale dei registri nella memoria RAM. Forzate quindi con POKE i valori che dovete porre nei registri nelle posizioni di memoria 69 per il registro A, 70 per il registro X e 71 per il registro Y. Se ora eseguite una CALL -193 questa subroutine trasferisce i tre valori dalla RAM nei registri A, X e Y. Potete infine eseguire la subroutine che vi interessa.

In Applesoft dovete procedere diversamente. In questo secondo caso dovete scrivere ed eseguire una subroutine che carica i registri con i valori voluti e quindi esegue una istruzione JSR di salto alla subroutine voluta.

Tabella D-1 - Subroutine di Sistema elencate per funzione

	Funzione	Punto d'inizio
Grafica in bassa risoluzione	Traccia un punto grafico in bassa risoluzione	$F800
	Traccia una linea orizzontale in bassa risoluzione	$F819
	Traccia una linea verticale in bassa risoluzione	$F828
	Pulisce e porta in nero tutte le 48 linee grafiche in bassa risoluzione (se in modo testo visualizza "@" inverso)	$F832
	Pulisce e porta in nero le prime 40 linee in bassa risoluzione (oppure le riempie con "@" inverso)	$F836
	Incrementa di tre l'attuale colore in bassa risoluzione	$F85F
	Impone un colore in bassa risoluzione	$F864
	Legge il colore di un punto grafico in bassa risoluzione	$F871
	Impone il modo grafico in bassa risoluzione, pulisce lo schermo e apre la finestra di testo di quattro linee	$FB40
Ingresso	Attende la battuta di un tasto mentre il cursore lampeggia e pone quindi il seme del generatore dei numeri a caso agli indirizzi $4E e $4F (78 e 79)	$FD1B
	Come la precedente, ma sono consentiti anche i codici "escape"	$FD35
	Invia un ritorno del carrello allo schermo e quindi permette l'ingresso di una linea lunga sino a 256 caratteri	$FD67
Uscita	Invia tre spazi vuoti alla periferica di uscita selezionata	$F948
	Invia da uno a 256 spazi vuoti alla periferica di uscita selezionata	$F94A
	Invia allo schermo un ritorno carrello ed un avanzamento di linea	$FC62
	Invia un carattere alla periferica di uscita selezionata	$FDED
	Invia un carattere alla finestra di testo	$FDF0

Tabella D-1 - Subroutine di Sistema elencate per funzione (continuo)

	Funzione	**Punto d'inizio**
Uscita campanello	Invia il carattere BELL (campanello – codice ASCII 7) alla periferica di uscita attualmente selezionata	$FBD9
	Genera un "beep" di durata 1/10 di secondo	$FBE4
	Fa stampare il messaggio ERR e suonare un "beep"	$FF20
	Fa suonare un "beep"	$FF3A
Finestra di testo	Impone sullo schermo 24 righe e 40 colonne	$FB2F
	Fa scorrere (scroll) la finestra di testo di una riga verso l'alto	$FC70
Controllo del cursore	Invia allo schermo un carattere di spostamento all'indietro e riaggiusta la posizione del cursore	$FC10
	Muove il cursore verso l'alto di una riga; se il cursore è già sulla prima linea non si muove	$FC1A
	Muove il cursore verso il basso di una riga sulla stessa colonna; se il cursore si trova sulla riga inferiore dello schermo viene eseguito uno "scroll" verso l'alto della finestra di testo	$FC66
Pulitura dello schermo	Pulisce la finestra di testo dalla posizione del cursore sino all'angolo in basso di destra	$FC42
	Pulisce la finestra di testo dal punto, le cui coordinate sono passate tramite i registri, all'angolo in basso a destra	$FC46
	Pulisce tutto lo schermo di testo e porta il cursore nell'angolo in alto a sinistra	$FC58
	Pulisce una linea di testo dal cursore alla sua estremità di destra	$FC9C
Video	Impone il modo video inverso (nero in campo bianco)	$FE80
	Impone il modo video normale (bianco in campo nero)	$FE84

Tabella D.1 — Subroutine di Sistema elencate per funzione. (continuo)

	Funzione	**Punto d'inizio**
Stampa contenuto dei registri	Stampa il contenuto dei registri Y e X con il formato YYXX sulla periferica di uscita selezionata	$F940
	Stampa il contenuto dei registri A e X con il formato AAXX sulla periferica di uscita selezionata	$F941
	Stampa il contenuto del registro X sulla periferica di uscita selezionata	$F944
	Stampa il contenuto del registro A sulla periferica di uscita selezionata	$FDDA
Muove il contenuto dei registri	Ripristina il contenuto dei registri (solo se la routine $FF4A è stata eseguita precedentemente)	$FF3F
	Salva il contenuto dei registri nelle posizioni di memoria della Pagina Zero: Reg. A in 69 ($45) — Reg. S in 72 ($48) — Reg. X in 70 ($46) — Reg. Y in 71 ($47) — Stack Pointer in 73 ($49)	$FF4A
Varie	Legge lo stato di un "paddle"	$FB1E
	Esegue un ciclo di ritardo: $0.5(5x^2 + 27x + 26)$ microsec.	$FCA8
	Fa ritornare al BASIC, cancellando il programma e le variabili in memoria	$FEBO
	Punto d'ingresso al Monitor	$FF69

Tabella D.2 – Subroutine di Sistema elencate per punto d'inizio.

Punto d'inizio	**Funzione**	**Registri da caricare prima della chiamata**	**Registri coinvolti**	**Equivalenza BASIC**
$F800	Traccia un punto grafico in bassa risoluzione	Porre la riga in A e la colonna in Y	Nessuno	PLOT
$F819	Traccia una linea orizzontale in bassa risoluzione	La riga in A, la colonna sinistra in Y e la colonna di destra all'indirizzo 44	A e Y	HLIN
$F828	Traccia una linea verticale in bassa risoluzione	La colonna in Y, la riga alta in A e la riga bassa all'indirizzo 45	Nessuno	VLIN
$F832	Pulisce e porta in nero tutte le 48 linee grafiche in bassa risoluzione (se in modo testo visualizza "@" inverso)	Nessuno	A e Y	CALL – 1998
$F836	Pulisce e porta in nero le prime 40 linee in bassa risoluzione (oppure le riempie con "@" inverso)	Nessuno	A e Y	GR (vedere $FB40)
$F85F	Incrementa di tre l'attuale colore in bassa risoluzione	Nessuno	A	CALL – 1985
$F864	Impone un colore in bassa risoluzione	Numero di colore in A	A	COLOR
$F871	Legge il colore di un punto grafico in bassa risoluzione	Riga in A e colonna in Y	A contiene il numero di colore	SCRN
$F940	Stampa il contenuto dei registri Y e X con il formato YYXX sulla periferica di uscita selezionata	Nessuno	Nessuno	CALL – 1728
$F941	Stampa il contenuto dei registri A e X con il formato AAXX sulla periferica di uscita selezionata	Nessuno	Nessuno	CALL – 1727
$F944	Stampa il contenuto del registro X sulla periferica di uscita selezionata	Nessuno	Nessuno	CALL – 1724
$F948	Invia tre spazi vuoti alla periferica di uscita selezionata	Nessuno	X e A	CALL – 1720
$F94A	Invia da uno a 256 spazi vuoti alla periferica di uscita selezionata	Numero di spazi vuoti in A (0 fa stampare 256 spazi)	Nessuno	SPC ( )[A] CALL – 1718
$FB1E	Legge lo stato di un "paddle"	Il numero del "paddle" in X	0-FF in Y; il contenuto di A è perso	PDL( )
$FB2F	Impone sullo schermo 24 righe e 40 colonne	Nessuno	A	TEXT
$FB40	Impone il modo grafico in bassa risoluzione, pulisce lo schermo e apre la finestra di testo di quattro linee	Nessuno	A e Y	GR
$FBD9	Invia il carattere BELL (campanello – codice ASCII 7) alla periferica di uscita attualmente selezionata	Nessuno	A e Y	CALL – 1063
$FBE4	Genera un "beep" di durata 1/10 di secondo	Nessuno	A e Y	CALL – 1052
$FC10	Invia allo schermo un carattere di spostamento all'indietro e riaggiusta la posizione del cursore	Nessuno	A	CALL – 1008
$FC1A	Muove il cursore verso l'alto di una riga; se il cursore è già sulla prima linea non si muove	Nessuno	A	CALL – 998
$FC42	Pulisce la finestra di testo dalla posizione del cursore sino all'angolo in basso di destra	Nessuno	A e Y	CALL – 958
$FC46	Pulisce la finestra di testo dal punto, le cui coordinate sono passate tramite i registri, all'angolo in basso a destra	La colonna in Y e la riga in A	A e Y	CALL – 954
$FC58	Pulisce tutto lo schermo di testo e porta il cursore nell'angolo in alto a sinistra	Nessuno	A e Y	HOME[A] CALL – 936
$FC62	Invia allo schermo un ritorno carrello ed un avanzamento di linea	Nessuno		CALL – 926

[A] È un comando BASIC disponibile solo in Applesoft.

Punto d'inizio	Funzione	Registri da caricare prima della chiamata	Registri coinvolti	Equivalenza BASIC
\$FC66	Muove il cursore verso il basso di una riga sulla stessa colonna; se il cursore si trova sulla riga inferiore dello schermo viene eseguito uno "scroll" verso l'alto della finestra di testo	Nessuno	A e Y	CALL – 922
\$FC70	Fa scorrere (scroll) la finestra di testo di una riga verso l'alto	Nessuno	A e Y	CALL – 912
\$FC9C	Pulisce una linea di testo dal cursore alla sua estremità di destra	Nessuno	A e Y	CALL – 868
\$FCA8	Esegue un ciclo di ritardo: 0.5 $(5x^2 + 27x + 26)$ microsec.	Il valore di ritardo x in A	A	CALL – 856
\$FD1B	Attende la battuta di un tasto mentre il cursore lampeggia e pone quindi il seme del generatore dei numeri a caso agli indirizzi \$4E e \$4F (78 e 79)	Nessuno	In A il carattere battuto; X e Y	CALL – 756
\$FD35	Come la precedente, ma sono consentiti anche i codici "escape"	Nessuno	In A il carattere battuto; X e Y	CALL – 715
\$FD67	Invia un ritorno del carrello allo schermo e quindi permette l'ingresso di una linea lunga sino a 256 caratteri	Il carattere di pronto all'indirizzo 51	Y e A; X contiene il numero di caratteri in ingresso che sono memorizzati dalla posizione \$200	INPUT
\$FDDA	Stampa il contenuto del registro A sulla periferica di uscita selezionata	Il dato in A	A	CALL – 550
\$FDED	Invia un carattere alla periferica di uscita selezionata	Il carattere in A	Nessuno	PRINT
\$FDFO	Invia un carattere alla finestra di testo	Il carattere in A	Nessuno	PRINT
\$FE8O	Impone il modo video inverso (nero in campo bianco)	Nessuno	Y	INVERSE[A] CALL – 384
\$FE84	Impone il modo video normale (bianco in campo nero)	Nessuno	Y	NORMAL[A] CALL – 380
\$FEBO	Fa ritornare al BASIC, cancellando il programma e le variabili in memoria	Nessuno	A, X e Y	CALL – 336
\$FF2O	Fa stampare il messaggio ERR e suonare un "beep"	Nessuno	A	CALL – 211
\$FF3A	Fa suonare un "beep"	Nessuno	A	CALL – 198
\$FF3F	Ripristina il contenuto dei registri (solo se la routine \$FF4A è stata eseguita precedentemente)	Nessuno	I contenuti dei registri sono ripristinati da queste posizioni: Reg. A : 69 (\$45) – Reg. S : 72 (\$48) – Reg. X : 70 (\$46) – Reg. Y : 71 (\$47) – Stack Pointer: 73 (\$49)	CALL – 193
\$FF4A	Salva il contenuto dei registri nelle posizioni di memoria della Pagina Zero: Reg. A in 69 (\$45) – Reg. S in 72 (\$48) – Reg. X in 70 (\$46) – Reg. Y in 71 (\$47) – Stack Pointer in 73 (\$49)	Nessuno	Nessuno	CALL – 182
\$FF69	Punto d'ingresso al Monitor	Nessuno	Nessuno	CALL – 151

APPENDICE E

INDIRIZZI UTILI DI PEEK E POKE

Gli indirizzi di memoria sino a 32767 sono espressi con lo stesso numero positivo decimale. Gli indirizzi superiori a 32767 sono espressi invece come numero negativo. In questo secondo caso per ottenere il valore positivo equivalente dovete sommare 65536 (per es. 49152 = 65536 – 16384).

Alcune delle funzioni, qui di seguito descritte, si attuano semplicemente richiamandole. Per esempio PEEK accede direttamente alla posizione indicata e svolge la sua funzione. POKE invece, a causa delle caratteristiche del microprocessore contenuto nell'Apple II, esegue la sua funzione *due volte*. In un certo senso POKE *equivale* a due PEEK. Generalmente questo fatto può essere trascurato, ma nel caso come – 16336 ("click" dell'altoparlante) si nota la differenza.

FINESTRA DI TESTO E INDIRIZZI DI CONTROLLO DEL CURSORE

32 Margine sinistro della finestra di testo

Indica la colonna del margine sinistro. PEEK ritorna un valore compreso tra 0 e 39 (0 è la prima colonna di sinistra dello schermo). Cambiando il valore di questa posizione, non si influenza la larghezza della finestra; si muovono invece ambedue i margini destro e sinistro. Se con POKE imponete in questo indirizzo un valore superiore a 39, o se questo valore più la larghezza della finestra supera 40, alcuni dei dati destinati allo schermo possono essere posti fuori dell'area di memoria dello schermo distruggendo una zona di programma.

33 Larghezza della finestra di testo

Indica la larghezza della finestra di testo. Il valore può essere compreso tra 1 e 40. Con questo valore si definisce il margine destro della finestra. La colonna del margine destro è ottenuta come somma del valore contenuto nell'indirizzo 32 e la larghezza della finestra (indirizzo 33).

Se diamo un valore di 0 (cioè larghezza zero) si può distruggere l'interprete BASIC. Se si impone che la colonna del margine destro sia superiore a 40, parte dei testi destinati allo schermo andranno a finire in altra zona della memoria. In tal caso si può rovinare il programma o una parte del software operativo.

34 Margine alto della finestra di testo

Indica la riga in alto della finestra di testo. Il valore può essere compreso tra 0 e 23. 0 è la riga in alto dello schermo, 23 quella in basso. Se imponete un valore superiore a 23, parte, o tutti, i dati inviati alla finestra andranno in una zona di memoria destinata ad altri scopi causando quindi dei danni. Non ponete mai il margine alto della finestra più in basso della riga inferiore dello schermo.

35 Margine basso della finestra di testo

Indica la riga inferiore della finestra di testo. Il valore può essere compreso tra 0 e 23. 0 è la riga in alto dello schermo, 23 quella in basso. Se imponete un valore superiore a 23, parte o tutti i dati inviati alla finestra andranno in una zona di memoria destinata per altri scopi causando quindi dei danni. Non ponete mai il margine basso della finestra sopra quello alto.

36 Posizione orizzontale del cursore

Specifica la posizione orizzontale attuale del cursore. PEEK ritorna un valore compreso tra 0 e 39 e indica la posizione del cursore relativa al margine sinistro della finestra (non necessariamente il margine sinistro dello schermo). Questo indirizzo può essere usato per posizionarsi oltre il margine destro della finestra (e quindi scrivere con PRINT), ma il cursore vi rimane solo per scrivere un unico carattere. Non ponete in questo indirizzo un valore, che sommato al margine sinistro (indirizzo 32), superi 39.
In questo caso PEEK equivale alla funzione dell'Applesoft POS.

37 Posizione verticale del cursore

Specifica la posizione verticale attuale del cursore. PEEK ritorna un valore tra 0 e 23 relativo alla cima dello schermo (non la cima della finestra). Non inserite in questo indirizzo un valore superiore a 23.

INDIRIZZI DEGLI ERRORI

216 Errore di flag

Indica se è in atto un ONERR GOTO. Se il bit 7 di questa posizione è 1 (cioè se il valore è maggiore o eguale a 128), vuol dire che è stata incontrata una istruzione ONERR GOTO. Se si verifica un errore il programma prosegue alla linea indicata da ONERR GOTO. Per disabilitare l'istruzione ONERR GOTO potete forzare un numero inferiore a 128 con POKE

218 - 219 Indicazione del numero di linea di un errore

Quando un errore causa un salto con ONERR GOTO, questi due indirizzi riportano il numero di linea ove si è verificato l'errore. Il numero di linea è dato da PEEK (219) * 256 + PEEK (218).

222 Codice del tipo di errore

Specifica il tipo di errore. Vedere l'Appendice C per il codice degli errori.

INDIRIZZI PER LA TASTIERA

- 16384 Carattere dalla tastiera
  Legge la tastiera. Se il valore in questo indirizzo è maggiore di 127 (cioè il bit 7 è 1) significa che un tasto è stato premuto. Si può calcolare il valore ASCII dell'ultimo tasto premuto sottraendo 128 dal contenuto di questo indirizzo.
- 16368 Flag della tastiera
  Azzera il bit 7 dell'indirizzo –16384 così che possa essere letto il prossimo carattere.

INDIRIZZI DI USCITA DEL "CLICK"

- 16352 Click della cassetta
  Genera un click udibile sul connettore jack della cassetta.
- 16336 Click dell'altoparlante
  Genera un click nell'altoparlante interno.

INTERRUTTORI DELLO SCHERMO

Gli "interruttori" che qui riportiamo non sono dei veri interruttori elettrici, ma degli indirizzi di memoria che determinano le caratteristiche dello schermo. Questi interruttori possono essere "aperti" o "chiusi" mediante le funzioni PEEK o POKE. In Figura E.1 sono riportati quattro interruttori che possono assumere valori diversi. Se viene selezionato il modo testo, l'unico altro interruttore che ha effetto è quello che cambia la pagina 1 con la pagina 2.

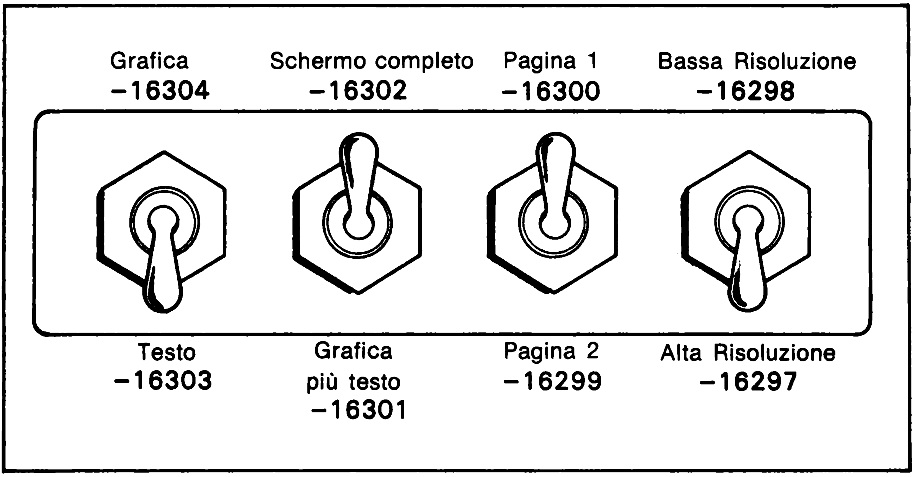


Figura E.1 — Indirizzi PEEK/POKE per grafici e testi.

– 16304 Seleziona il modo grafico
Seleziona il modo grafico. Lo schermo non viene però pulito e portato tutto in nero. Gli altri tre interruttori possono assumere qualunque valore.

– 16303 Seleziona il modo testo
Seleziona il modo testo. Può essere scelta sia la pagina 1 che quella 2.

– 16302 Seleziona il modo grafico con schermo completo
Seleziona il modo grafico con lo schermo completo; questo non risulterà però visibile sino a che non vi sia accesso all'indirizzo –16304.

– 16301 Seleziona il modo grafico con una finestra di testo
Abilita la finestra di testo di quattro righe sulla parte bassa dello schermo. Se lo schermo è già in modo testo, questa finestra non sarà visibile sino a che non vi sia accesso all'indirizzo –16304.

– 16300 Seleziona la pagina 1
Seleziona la pagina 1, o grafica o di testo.

– 16299 Seleziona la pagina 2
Seleziona la pagina 2, o grafica o di testo.

– 16298 Seleziona la grafica in bassa risoluzione
Seleziona la grafica in bassa risoluzione. Se lo schermo è in modo testo non avrà effetto sino a che non vi sia accesso all'indirizzo –16304.

– 16297 Seleziona la grafica in alta risoluzione
Seleziona la grafica in alta risoluzione. Se lo schermo è in modo testo non avrà effetto sino a che non vi sia accesso all'indirizzo –16304.

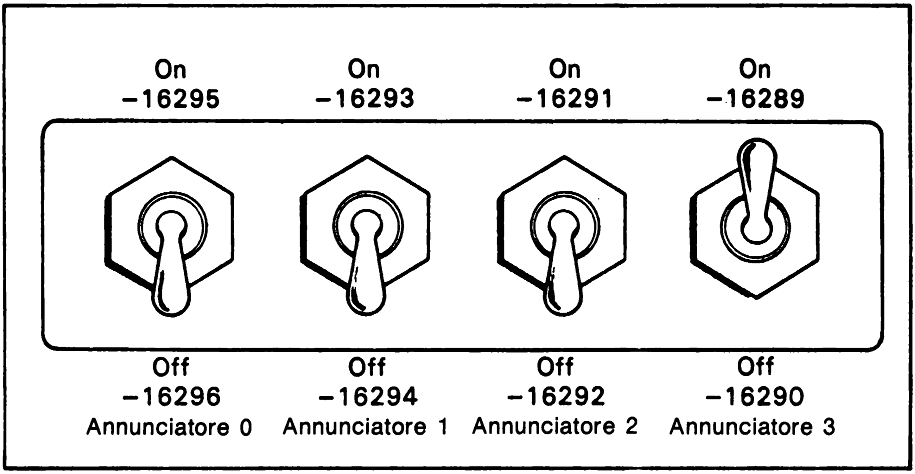


Figura E.2 – Indirizzi delle uscite dei controlli dei giochi (Annunciatori).

INDIRIZZI PER CONTROLLO DEI GIUOCHI

Questi indirizzi accendono e spengono le uscite di controllo dei giuochi, segnalano se i pulsanti sono, o non sono, abbassati e generano un segnale di "strobe". La Figura E.2 riporta schematicamente questi interruttori. In Figura E.3 è riportato il connettore generale dei giuochi.

– 16296 Annunciatore 0 spento
  Spegne l'uscita di controllo giuochi (annunciatore) numero 0. La tensione sul pin 15 del connettore dei giuochi è portata circa a 0 volt (TTL alto).

– 16295 Annunciatore 0 acceso
  Accende l'uscita di controllo giuochi (annunciatore) numero 0. La tensione sul pin 15 del connettore dei giuochi è portata a circa +5volt (TTL basso).

– 16294 Annunciatore 1 spento
  Spegne l'uscita di controllo giuochi (annunciatore) numero 1. La tensione sul pin 14 del connettore dei giuochi è portata a circa 0 volt (TTL alto).

– 16293 Annunciatore 1 acceso
  Accende l'uscita di controllo giuochi (annunciatore) numero 1. La tensione sul pin 14 del connettore dei giuochi è portata a circa +5 volt (TTL basso).

– 16292 Annunciatore 2 spento
  Spegne l'uscita di controllo giuochi (annunciatore) numero 2. La tensione sul pin 13 del connettore dei giuochi è portata a circa 0 volt (TTL alto).

– 16291 Annunciatore 2 acceso
  Accende l'uscita di controllo giuochi (annunciatore) numero 2. La tensione sul pin 13 del connettore dei giuochi è portata a circa +5 volt (TTL basso).

– 16290 Annunciatore 3 spento
  Spegne l'uscita di controllo giuochi (annunciatore) numero 3. La tensione sul pin 12 del connettore dei giuochi è portata a circa 0 volt (TTL alto).

– 16289 Annunciatore 3 acceso
  Accende l'uscita di controllo giuochi (annunciatore) numero 3. La tensione sul pin 12 del connettore dei giuochi è portata a circa +5 volt (TTL basso).

– 16287 Lettura del pulsante 0
  Quando il pulsante del controllore 0 viene premuto, in questo indirizzo viene forzato un valore superiore a 127; diversamente esso contiene un valore eguale o inferiore a 127. Il pulsante 0 è connesso al pin 2 del connettore dei giuochi.

– 16286 Lettura del pulsante 1
  Quando il pulsante del controllore 1 viene premuto, in questo indirizzo viene formato un valore superiore a 127; diversamente esso contiene un valore eguale o inferiore a 127. Il pulsante 1 è connesso al pin 3 del connettore dei giuochi.

— 16285 Lettura del pulsante 2

Quando il pulsante del controllore 2 viene premuto, in questo indirizzo viene forzato un valore superiore a 127; diversamente esso contiene un valore eguale o inferiore a 127. Il pulsante 2 è connesso al pin 4 del connettore dei giuochi.

— 16272 Uscita dello strobe

Normalmente il pin 5 del connettore dei giuochi è tenuto a +5 volt. Se date un comando PEEK all'indirizzo —16285 esso scende a 0 volt per mezzo microsecondo. Se date invece POKE questo si ripete due volte.

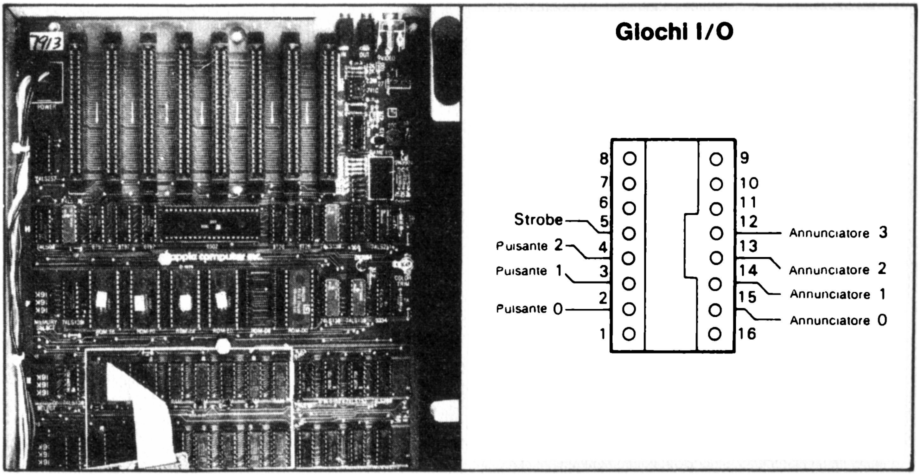


Figura E.3 — Ingressi e uscite del controllo dei giochi.

APPENDICE F

PAROLE RISERVATE DEL BASIC

Il calcolatore Apple II, quando incontra una delle seguenti parole, le interpreta come comandi, o istruzioni, o funzioni del BASIC. Solo se queste parole sono poste tra le virgolette di un testo, allora non assumono alcun significato di comando. Fate molta attenzione quindi a non usare queste parole come parti di nomi di variabili, specialmente nel caso delle parole riservate molto corte.

La presenza di spazi inseriti nelle parole riservate non ne cambia il significato; l'Apple II effettua automaticamente l'elisione degli spazi vuoti.

INTEGER BASIC

ABS	END	LET	PDL	SAVE
AND	FOR	LIST	PEEK	SCRN
ASC	GOSUB	LOAD	PLOT	SGN
AT	GOTO	LOMEM:	POKE	STEP
AUTO	GR	MAN	POP	TAB
CALL	HIMEM:	MOD	PRINT	TEXT
COLOR=	HLIN	NEW	PR#	THEN
CON	IF	NEXT	REM	TO
DEL	IN#	NOT	RETURN	TRACE
DIM	INPUT	NOTRACE	RND	VLIN
DSP	LEN	OR	RUN	VTAB

APPLESOFT

Le parole riservate dell'Applesoft sono automaticamente convertite in un codice di un byte. In questa Appendice le parole sono riportate in ordine alfabetico, mentre nell'Appendice I sono riportate in ordine numerico.

L'Applesoft non riconosce correttamente la parola TO se:

1) Il primo carattere non di spazio, prima di TO, è la lettera A e

2) Uno o più spazi separano la T dalla O.

ABS	(212)	HTAB	(150)	REM	(178)
AND	(205)	IF	(173)	RESTORE	(174)
ASC	(230)	IN#	(139)	RESUME	(166)
AT	(197)	INPUT	(132)	RETURN	(177)
ATN	(225)	INT	(211)	RIGHT$	(233)
CALL	(140)	INVERSE	(158)	RND	(219)
CHR$	(231)	LEFT$	(232)	ROT=	(152)
CLEAR	(189)	LEN	(227)	RUN	(172)
COLOR=	(160)	LET	(170)	SAVE	(183)
CONT	(187)	LIST	(188)	SCALE=	(153)
COS	(222)	LOAD	(182)	SCRN(	(215)
DATA	(131)	LOG	(220)	SGN	(210)
DEF	(184)	LOMEM:	(164)	SHLOAD	(154)
DEL	(133)	MID$	(234)	SIN	(223)
DIM	(134)	NEW	(191)	SPC(	(195)
END	(128)	NEXT	(130)	SPEED=	(169)
EXP	(221)	NORMAL	(157)	SQR	(218)
FLASH	(159)	NOT	(198)	STEP	(199)
FN	(194)	NOTRACE	(156)	STOP	(179)
FOR	(129)	ON	(180)	STORE	(168)
FRE	(214)	ONERR	(165)	STR$	(228)
GET	(190)	OR	(206)	TAB(	(192)
GOSUB	(176)	PDL	(216)	TAN	(224)
GOTO	(171)	PEEK	(226)	TEXT	(137)
GR	(136)	PLOT	(141)	THEN	(196)
HCOLOR=	(146)	POKE	(185)	TO	(193)
HGR	(145)	POP	(161)	TRACE	(155)
HGR2	(144)	POS	(217)	USR	(213)
HIMEM:	(163)	PRINT	(186)	VAL	(229)
HLIN	(142)	PR#	(138)	VLIN	(143)
HOME	(151)	READ	(135)	VTAB	(162)
HPLOT	(147)	RECALL	(167)	WAIT	(181)
				XDRAW	(149)

DOS

I comandi del DOS sono considerati parole riservate solo se sono usate in modo immediato oppure in una PRINT che inizia con CTRL-D (codice ASCII 4).

APPEND	CHAIN	INIT	POSITION	SAVE
BLOAD	CLOSE	LOAD	READ	UNLOCK
BRUN	DELETE	LOCK	RENAME	VERIFY
BSAVE	EXEC	OPEN	RUN	WRITE

APPENDICE G

IMPIEGO DELLA MEMORIA

ORGANIZZAZIONE GENERALE DELLA MEMORIA

La memoria del calcolatore Apple II è divisa in tre parti principali: la memoria RAM di lettura e scrittura (detta anche ad accesso diretto o random), la memoria a sola lettura ROM e la memoria di ingresso/uscita I/O.

Gli indirizzi da 0 a 49151 ($BFFF in esadecimale) sono RAM, gli indirizzi da 49152 a 53247 ($CFFF) sono ROM. È ovvio però che non tutti i sistemi abbiano la stessa quantità di memoria. Nel caso di memorie più piccole una parte della RAM non è accessibile. Per esempio se avete 16K di RAM, gli indirizzi da 16384 ($4000) a 49151 ($BFFF) non sono usabili.

Nella Tabella G.1 viene riportato come è organizzata la memoria di un calcolatore Apple II. Notate che vi sono due blocchi di memoria libera vicini alle due pagine grafiche in alta risoluzione. Il puntatore LOMEM: punta all'estremo inferiore dell'area libera, mentre il puntatore HIMEM: punta all'estremo superiore. Questa zona di memoria può essere usata per diversi scopi come per fare risiedere l'interprete Applesoft, proveniente da cassetta o da disco, oppure il sistema operativo a disco, oppure la grafica in alta risoluzione oppure ancora i vostri programmi BASIC.

Tabella G.1 — Organizzazione della memoria BASIC.

Posizioni		**Tipo di Memoria**	**Funzione**
Decimale	**Esadecimale**		
0-255	$0-$0FF	RAM	Programmi di Sistema
256-511	$100-$1FF	RAM	Stack del sistema
512-767	$200-$2FF	RAM	Buffer di ingresso della tastiera
768-1023	$300-$3FF	RAM	Posizioni dei vettori del Monitor
1024-2047	$400-$7FF	RAM	Pagina 1 di testc e grafica in bassa risoluzione
2048-3071	$800-$BFF	RAM	Pagina 2 di testo e grafica in bassa risoluzione
3072-8191	$C00-$1FFF	RAM	Zona libera
8192-16383	$2000-$3FFF	RAM	Pagina grafica in alta risoluzione 1
16384-24575	$4000-$5FFF	RAM	Pagina grafica in alta risoluzione 2
24576-49151	$6000-$BFFF	RAM	Zona libera
49152-49279	$C000-$C07F	I/O	Posizioni riservate
49280-49407	$C080-$C0FF	I/O	Spazio per cartoline I/O
49408-51199	$C100-$C7FF	I/O	Spazio per cartoline di espan-
51200-53247	$C800-$CFFF	I/O	sione della memoria
53248-65535	$D000-$FFFF	ROM	Integer BASIC, Applesoft, Monitor o Autostart Monitor, ecc.

GLI INTERPRETI DEL BASIC

Come potete vedere nella Tabella G.1, l'interprete Integer BASIC risiede sempre nella memoria ROM. L'interprete Applesoft risiede nella memoria ROM solo se avete la cartolina firmware oppure la cartolina Language System. Diversamente l'Applesoft occupa circa 10K di memoria a partire dall'indirizzo 2048 ($800).

MEMORIA OCCUPATA DAL DOS

Per potere usare il DOS avete bisogno di un calcolatore con almeno 16K di memoria. Il DOS, quando viene caricato, occupa circa 10K nella parte alta della memoria. HIMEM: viene portato al punto appena inferiore all'area occupata dal DOS. In Figura G.1 vi mostriamo dove risiede il DOS nel caso di diverse ampiezze di memoria. Notate che avete bisogno di almeno 24K di memoria RAM per potere lavorare contemporaneamente con il DOS e con l'Applesoft proveniente da disco, o da cassetta, e di 32K per lavorare anche la pagina 1 della grafica in alta risoluzione. In Figura G.1 si vede anche, chiaramente, quando si crea un conflitto nell'occupazione della memoria tra le varie parti di software.

Durante il booting del DOS vengono impegnate anche altre zone della memoria come indicato nella Figura G.2. Ovviamente tutto ciò che era presente in queste aree, prima del booting, viene perso.

OCCUPAZIONE DELLA MEMORIA IN INTEGER BASIC

Le linee di un programma in Integer BASIC risiedono nella parte alta libera della memoria RAM a partire dall'indirizzo HIMEM:. Come mostrato nella Figura G.3 il valore HIMEM: viene automaticamente variato se voi aggiungete, o togliete, o cambiate delle linee di programma.

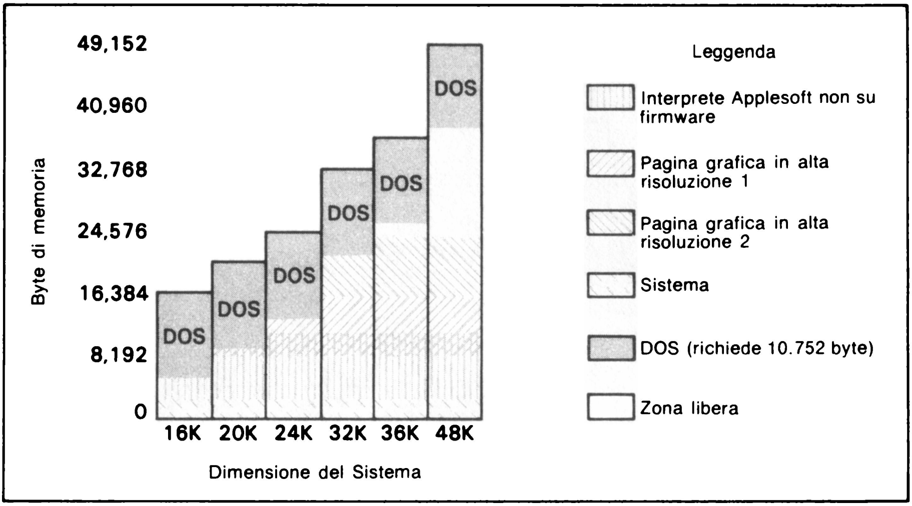


Figura G.1 — Impiego della memoria RAM di lettura e scrittura.

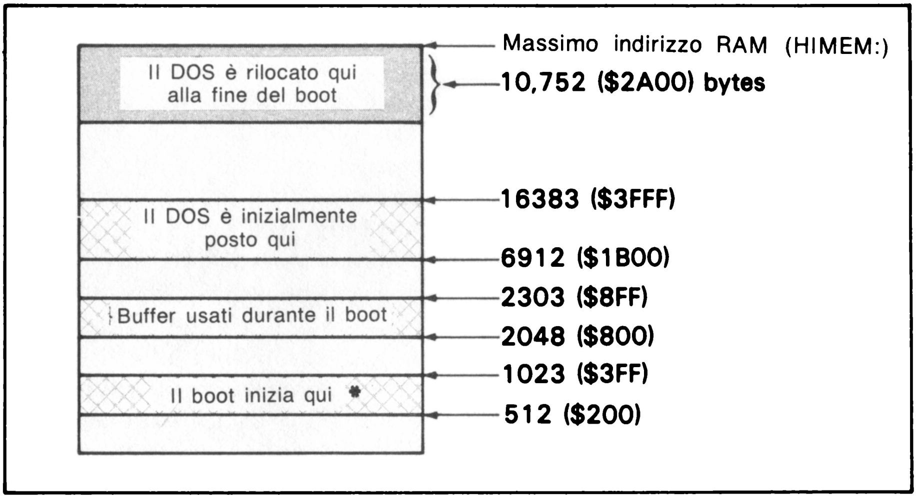


Figura G.2 — Impegno della memoria durante il "booting" del DOS.

Le variabili sono invece memorizzate a partire dall'indirizzo LOMEM:, ma in questo caso verso l'alto. Il valore LOMEM: viene automaticamente corretto con la presenza delle variabili. Ogni variabile numerica viene memorizzata in quattro parti: il nome della variabile, il byte DSP, (on-off), l'indirizzo di memoria della variabile successiva e il valore, o i valori, della variabile.

Il nome della variabile può essere lungo sino a 100 caratteri e ogni carattere viene memorizzato in ASCII con il bit di ordine più alto posto eguale a 1.

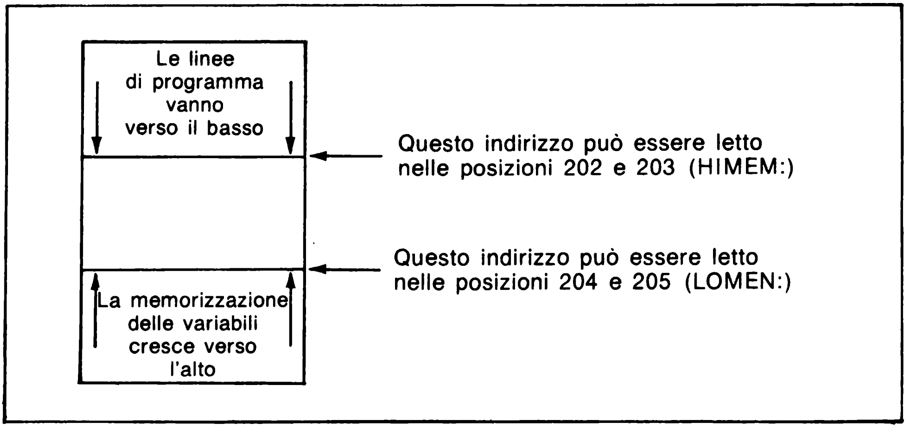


Figura G.3 — Mappa della memoria di un programma in Integer BASIC.

Il byte DSP indica se il comando DSP è attivo per questa variabile. Questo byte è tenuto normalmente a zero, ma viene portato a 1 se il comando DSP tiene sotto controllo la variabile.

L'indirizzo della variabile successiva viene memorizzato in due byte con il byte di ordine basso per primo.

Il dato viene memorizzato in due byte successivi (il byte di ordine basso per primo). Se la variabile non è un array, esiste quindi solo un dato. Se la variabile è un array, cioè ha un indice, tutti i suoi elementi sono riportati, con ordine, in coppie successive di byte a partire dall'elemento con indice zero. Non c'è quindi differenza tra una variabile senza indice e l'elemento zero di una variabile con indice, ma con lo stesso nome.

Le variabili di stringa sono memorizzate analogamente. Il nome della variabile, il byte DSP e l'indirizzo della variabile successiva sono memorizzati nello stesso modo delle variabili numeriche. In questo caso però ogni dato è un carattere che occupa un solo byte in codice ASCII. Anche in questo caso il bit di ordine più alto viene posto eguale a 1. L'ultimo carattere della striga è seguito da un byte di coda in cui il bit di ordine più alto è invece posto eguale a 0.

OCCUPAZIONE DELLA MEMORIA IN APPLESOFT

Le linee di un programma Applesoft occupano la parte bassa della memoria RAM libera. Come si vede nella Figura G.4 il puntatore LOMEM: si aggiorna automaticamente con la presenza, o la cancellazione, di nuove linee del programma. Le variabili numeriche semplici, e i puntatori delle stringhe, vengono memorizzati subito sopra le linee di un programma. Le variabili con indici e i puntatori, delle variabili con indici di stringa, sono memorizzati sopra le variabili semplici. I valori delle stringhe sono invece memorizzati nella parte alta della memoria a partire da HIMEM:. Man mano che usate più stringhe il valore di HIMEM: scende automaticamente.

Le variabili numeriche, e i puntatori di stringa, usano sette byte di memoria. Ogni variabile reale usa due byte, in codice ASCII, per il nome della variabile (ambedue con il bit di ordine alto posto eguale a 0). Il valore è invece memorizzato in cinque byte in forma di notazione scientifica; un byte per l'esponente e quattro byte per la mantizza (nell'ordine dal byte più significativo a quello meno significativo).

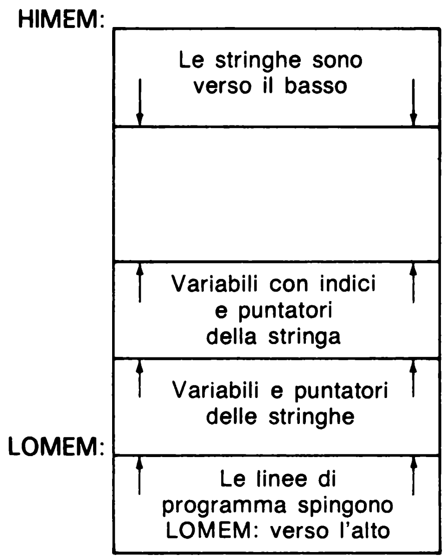


Figura G.4 — Mappa della memoria di un programma in Applesoft.

Ogni variabile intera usa due byte per codificare in ASCII il suo nome (ambedue hanno però il bit di ordine più alto eguale a 1). Il valore viene posto in due byte successivi con il byte di ordine più alto per primo.

Ogni puntatore di stringa usa due caratteri ASCII per il nome della variabile (il bit di ordine alto del primo è posto eguale a 1, mentre quello del secondo è eguale a ze-

ro). Successivamente c'è un byte per la lunghezza della stringa e due byte che puntano alla stringa vera e propria (il byte di ordine basso per primo). Gli ultimi tre byte di una variabile intera e gli ultimi due di una variabile di stringa non sono usati.

Le variabili con indici numeriche e di stringa sono memorizzate immediatamente sopra le variabili semplici. Il nome di queste variabili è memorizzato con due caratteri ASCII. Nel caso delle variabili reali ambedue i bit di ordine alto, dei due byte del nome, sono eguali a 0; le variabili intere li hanno invece eguali a 1, mentre per le variabili di stringa il primo bit di ordine alto è eguale a 1 e il secondo è eguale a 0.

Subito dopo il nome della variabile seguono i due byte che indicano l'indirizzo della variabile successiva. Questo indirizzo si riferisce al primo byte del nome della nuova variabile. Segue quindi il byte che indica il numero delle dimensioni. Successivamente coppie separate di byte (sempre prima il byte di ordine alto) indicano ognuna l'ampiezza delle singole dimensioni. Attenzione però che queste ampiezze delle dimensioni sono riportate in ordine inverso a quello degli indici della variabile.

Dopo questa descrizione della variabile con indici, seguono gli elementi della variabile a partire da quello (0, 0, ... 0) verso quello (N, N, ... N). Gli indici vengono incrementati iniziando da quello più a sinistra e poi via via quelli verso destra.

Ogni elemento reale occupa cinque byte, uno per l'esponente e quattro per la mantissa (il byte più significativo per primo). Gli elementi interi occupano invece due byte ognuno (con il byte di ordine più alto per primo). I puntatori degli elementi di stringa vogliono tre byte: uno per la lunghezza della stringa e due per l'indirizzo (byte di ordine basso per primo) della stringa vera e propria.

I valori delle stringhe sono memorizzati nella parte alta della memoria RAM libera. Ogni carattere occupa un byte. Le stringhe non vengono duplicate, ma più puntatori puntano allo stesso valore. Quando si generano nuove stringhe il valore di HIMEM: viene spostato verso il basso.

Il comando FRE permette di fare una "pulizia" delle stringhe che non si usano più, perchè altrimenti rimarrebbero presenti in memoria. Ovviamente in tal caso HIMEM: viene riportato in alto.

APPENDICE H

CARATTERISTICHE DEL DISK II

I dati sono registrati su un dischetto, in un drive Disk II, su 35 *tracce* concentriche che sono numerate da 0 a 34 (da $0 a $22). Ogni traccia viene a sua volta ripartita in 16 segmenti detti *settori* che sono numerati da 0 a 15 (da $0 a $F). Ogni settore può contenere sino a 256 byte. In totale un dischetto può contenere 116.480 byte su 455 settori.

Il sistema DOS trasferisce i dati da e verso il dischetto un settore alla volta. Per queste operazioni usa due buffer in memoria di 256 byte ciascuno; un buffer è usato per la lettura e l'altro per la scrittura per ogni file aperto.

Ogni tipo di file (binario, testo o programma) ha il suo formato di registrazione sul disco. I file di testo sono registrati in codice ASCII, un byte per ogni carattere. Un byte zero segna la fine del file. Tutti i byte di un testo sono interpretati solo come testo.

I primi due byte, del primo settore di un file di programma BASIC, indicano la lunghezza del programma (il byte basso per primo). Il resto del file contiene il programma in codice ASCII. Nel caso dell'Applesoft le parole riservate sono codificate in un singolo carattere ASCII e non lasciate per esteso carattere per carattere. Nell'Appendice F sono riportate le parole riservate in ordine alfabetico e nell'Appendice I in ordine numerico.

I primi due byte, del primo settore di un file binario, indicano l'indirizzo di partenza del file così come era nella memoria RAM (il byte basso per primo). I due byte successivi contengono la lunghezza del file (il byte basso per primo). Il resto del file contiene i dati binari.

LA LISTA DELLE TRACCE E DEI SETTORI

Il DOS normalmente scrive il contenuto del buffer sul primo settore che trova libero. Questo significa che un file, composto da più settori, può risiedere su tanti punti

diversi del dischetto. Il DOS deve quindi tenere una lista delle tracce e dei settori che sono usati da un singolo file. Questa *lista delle tracce e dei settori* viene a sua volta registrata in appositi settori del dischetto.

Ogni settore della lista, delle tracce e dei settori, contiene un puntatore al settore successivo e può puntare sino a 122 settori costituenti un unico file.

Se il settore di un file non viene usato, il puntatore a lui, nella lista tracce/settori, assume il valore 0. Se un intero settore, all'inizio della lista tracce/settori, ha tutti puntatori 0, tale settore non viene registrato sul dischetto. Per esempio se il record numero 5000 è il primo ed unico record, di un file ad accesso diretto che ha il parametro L eguale a 256 (cioè un record per settore), saranno impiegati solo due settori sul dischetto. Uno per i dati del record numero 5000 e uno per il 41-esimo settore della lista tracce - settori.

I byte 0 e quelli tra 3 e 12 dei settori, della lista tracce/settori, non sono usati. I byte 1 e 2 contengono i numeri di traccia e di settore del prossimo settore della lista. Se questi byte vengono posti eguali a zero, ciò significa che la lista è terminata.

LA DIRECTORY

Il DOS usa la traccia 17 ($11) per la directory del dischetto. Per ogni file, la directory contiene il nome del file, il tipo di file, il numero di settori occupati dal file (modulo 256) e l'indirizzo della lista tracce/settori. Quasi tutta questa informazione viene visualizzata con il comando CATALOG.

Ogni settore di directory può contenere informazioni riguardanti sino a sette file. Una directory inizia al settore 15 della traccia 17 per proseguire poi con il settore 14 e così avanti sino al settore 1. Una directory può contenere dati relativi sino a 84 file.

I byte 0 e quelli tra 3 e 10, di ogni settore di una directory, non sono usati. I byte 1 e 2 contengono rispettivamente il numero di traccia e di settore del settore successivo della directory. Se ambedue questi byte sono eguali a zero non vi sono altri settori nella directory. I byte da 11 a 255 contengono i dati della directory riguardanti ogni file. Per ogni file sono utilizzati 35 byte. Il primo gruppo di 35 byte è posto tra i byte 11 e 45, il secondo tra i byte 46 e 80, ecc.

Nella Tabella H.1 è riportata la spiegazione dei 35 byte. La Tabella H.2 spiega come il tipo di file viene codificato nel byte 2.

Il settore 0 della traccia 17 non riguarda la directory, ma contiene informazioni su tutto il dischetto. Esso contiene lo stato di identificazione, la descrizione fisica del dischetto e la disponibilità dello spazio libero. La Tabella H.3 descrive il contenuto di questo importante settore chiamato *Tavola dei Contenuti di Volume* (*Volume Table of Contents*).

Ogni gruppo di quattro byte, dal byte 56 al byte 195, della Tavola contiene una mappa di disponibilità per ogni traccia del dischetto. Ogni mappa riporta quali settori, della corrispondente traccia, sono in uso e quali sono liberi. Un bit viene posto a 0 quando il corrispondente settore è occupato; è invece posto a 1 se il settore è libero. Nella Tabella H.4 viene riportata la mappa di disponibilità dei settori.

Tabella H.1 – Formato d'ingresso in una directory.

Numero relativo di byte	Contenuto dei byte
0	Numero di traccia della lista di tracce/settori del file. Viene cambiato in 255 quando il file è cancellato (il suo vecchio contenuto è salvato nel byte 34).
1	Numero di settore della lista di tracce/settori del file.
2	Tipo di file. Vedere Tab. H-2.
3-32	Nome del file in ASCII.
33	Numero dei settori (modulo 256) usati dal file.
34	Segno di fine. Normalmente è zero, ma viene cambiato con il contenuto del byte 0, quando il file è cancellato.

Tabella H.2 – Codifica del tipo di file in una directory.

Bit	Tipo di file
0	Il file è un programma in Integer BASIC se questo bit è 1.
1	Il file è un programma in Applesoft se questo bit è 1.
2	Il file è un file binario se questo bit è 1.
3-6	Riservati per ampiamenti futuri.
7	Se questo bit è 1 il file è protetto.
Se tutti i bit da 0 a 6 sono nulli, il file è un file di testo.	

Tabella H.3 – Tavola dei Contenuti di Volume (Settore 0, Traccia 17).

Byte	Descrizione
0	Non usato
1	Numero di traccia del primo settore della directory
2	Numero di settore del primo settore della directory
3	Numero di revisione del DOS
4-5	Non usato
6	Numero di volume del dischetto
7-38	Non usato
39	Numero massimo di coppie traccia/settore possibili in ogni settore della lista delle tracce/settori del file
40-47	Non usato
48-51	Maschera per le mappe di disponibilità dei settori
52	Numero di tracce del dischetto
53	Numero di settori del dischetto
54-55	Numero di byte per settore: byte di ordine basso in 54 e quello di ordine alto in 55
56-59	Mappa di disponibilità dei settori, traccia 0
60-63	Mappa di disponibilità dei settori, traccia 1
64-195	Mappa di disponibilità dei settori, tracce da 2 a 195
196-255	Non usato

Tabella H.4 — Mappa di disponibilità dei settori.

Byte	**Bit**	**Settore**
Primo	7	12
	6	11
	5	10
	4	9
	3	8
	2	7
	1	6
	0	5
Secondo	7	4
	6	3
	5	2
	4	1
	3	0
	2-0	non usato
Terzo	Tutti	non usato
Quarto	Tutti	non usato

APPENDICE I

CODICI DELLE PAROLE RISERVATE APPLESOFT

Nella prima tabella di questa Appendice sono riportati i codici ASCII da 1 a 95. Nel calcolatore Apple II i codici da 96 a 127 producono sullo schermo gli stessi caratteri dei codici tra 64 e 95, mentre su altre periferiche possono generare i corrispondenti caratteri minuscoli. Nessun tasto della tastiera produce un codice compreso tra 96 e 127.

I codici ASCII tra 128 e 255 sono equivalenti a quelli tra 0 e 127. Nessun tasto genera uno di questi codici.

Nella seconda tabella sono riportate le parole riservate dell'Applesoft. Ogni parola riservata occupa quindi un solo byte nel programma memorizzato, sia nella memoria RAM che sul disco.

In questa Appendice le parole riservate sono elencate in ordine numerico, nell'Appendice F sono invece riportate in ordine alfabetico.

Codice ASCII

Codice ASCII	Carattere visualizzato	Tasto	Codice ASCII	Carattere visualizzato	Tasto
0		CTRL-@	48	0	0
1		CTRL-A	49	1	1
2		CTRL-B	50	2	2
3		CTRL-C	51	3	3
4		CTRL-D	52	4	4
5		CTRL-E	53	5	5
6		CTRL-F	54	6	6
7	(campanello)	CTRL-G	55	7	7
8	(carrello indietro)	CTRL-H o ←	56	8	8
9		CTRL-I	57	9	9
10	(avanzamento linea)	CTRL-J	58	:	:
11		CTRL-K	59	;	;
12		CTRL-L	60	<	<
13	(ritorno carrello)	CTRL-M	61	=	=
14		CTRL-N	62	>	>
15		CTRL-O	63	?	?
16		CTRL-P	64	@	@
17		CTRL-Q	65	A	A
18		CTRL-R	66	B	B
19		CTRL-S	67	C	C
20		CTRL-T	68	D	D
21	(carrello in avanti)	CTRL-U o →	69	E	E
22		CTRL-V	70	F	F
23		CTRL-W	71	G	G
24	(cancella una linea)	CTRL-X	72	H	H
25		CTRL-Y	73	I	I
26		CTRL-Z	74	J	J
27		Esc	75	K	K
28		n.a.	76	L	L
29		CTRL-SHIFT-M	77	M	M
30		CTRL-^	78	N	N
31		n.a.	79	O	O
32	spazio	barra di spazio	80	P	P
33	!	!	81	Q	Q
34	"	"	82	R	R
35	#	#	83	S	S
36	$	$	84	T	T
37	%	%	85	U	U
38	&	&	86	V	V
39	'	'	87	W	W
40	(	(	88	X	X
41	)	)	89	Y	Y
42	*	*	90	Z	Z
43	+	+	91	[	n.a.
44	,	,	92	\	n.a.
45	-	-	93	]	SHIFT-M
46	.	.	94	^	^
47	/	/	95		n.a.

n.a. = non disponibile sulla tastiera dell'Apple II.

Parole riservate e loro codici in Applesoft

Codice	Parole riservate	Codice	Parole riservate	Codice	Parole riservate
128	END	164	LOMEM:	200	+
129	FOR	165	ONERR	201	-
130	NEXT	166	RESUME	202	*
131	DATA	167	RECALL	203	/
132	INPUT	168	STORE	204	^
133	DEL	169	SPEED=	205	AND
134	DIM	170	LET	206	OR
135	READ	171	GOTO	207	>
136	GR	172	RUN	208	=
137	TEXT	173	IF	209	<
138	PR#	174	RESTORE	210	SGN
139	IN#	175	&	211	INT
140	CALL	176	GOSUB	212	ABS
141	PLOT	177	RETURN	213	USR
142	HLIN	178	REM	214	FRE
143	VLIN	179	STOP	215	SCRN(
144	HGR2	180	ON	216	PDL
145	HGR	181	WAIT	217	POS
146	HCOLOR=	182	LOAD	218	SQR
147	HPLOT	183	SAVE	219	RND
148	DRAW	184	DEF	220	LOG
149	XDRAW	185	POKE	221	EXP
150	HTAB	186	PRINT	222	COS
151	HOME	187	CONT	223	SIN
152	ROT=	188	LIST	224	TAN
153	SCALE=	189	CLEAR	225	ATN
154	SHLOAD	190	GET	226	PEEK
155	TRACE	191	NEW	227	LEN
156	NOTRACE	192	TAB(	228	STR$
157	NORMAL	193	TO	229	VAL
158	INVERSE	194	FN	230	ASC
159	FLASH	195	SPC(	231	CHR$
160	COLOR=	196	THEN	232	LEFT$
161	POP	197	AT	233	RIGHT$
162	VTAB	198	NOT	234	MID$
163	HIMEM:	199	STEP		

APPENDICE J

TABELLE DI CONVERSIONE NUMERICA

Questa Appendice contiene le seguenti tabelle di conversione:

- da esadecimale a binario
- da esadecimale a decimale intero

Conversione esadecimale - binario

La tabella seguente permette la conversione tra i valori esadecimali tra O e OF e quelli binari tra 0000 e 1111.

Se avete un numero binario con molte cifre, raggruppatele quattro a quattro da destra verso sinistra e convertite ogni quattro bit in una cifra esadecimale. Aggiungete se necessario degli zeri per formare quattro bit nel raggruppamento più a sinistra. Ecco un esempio:

$$100101_2 = \underbrace{\underbrace{0010}_{2}\;\underbrace{0101}_{5}}_{25_{16}}{}_2$$

Viceversa per convertire un numero esadecimale in binario è sufficiente convertire cifra per cifra. Ecco un esempio:

$$37_{16} \rightarrow 3 \quad 7 \rightarrow \overbrace{0110}^{3}\;\overbrace{0111}^{7} \rightarrow \underbrace{0110\;0111}_{01100111_2}$$

Esadecimale	Binario
OO	0000
O1	0001
O2	0010
O3	0011
O4	0100
O5	0101
O6	0110
O7	0111
O8	1000
O9	1001
OA	1010
OB	1011
OC	1100
OD	1101
OE	1110
OF	1111

CONVERSIONE DI INTERI TRA DECIMALE ED ESADECIMALE

La tabella permette la conversione diretta tra numeri interi in forma esadecimale, nell'intervallo 0-FFF, e numeri interi in forma decimale, nell'intervallo 0-4095. Per valori superiori potete sommare gli elementi seguenti:

Esadecimale	Decimale	Esadecimale	Decimale
01 000	4 096	20 000	131 072
02 000	8 192	30 000	196 608
03 000	12 288	40 000	262 144
04 000	16 384	50 000	327 680
05 000	20 480	60 000	393 216
06 000	24 576	70 000	458 752
07 000	28 672	80 000	524 288
08 000	32 768	90 000	589 824
09 000	36 864	A0 000	655 360
0A 000	40 960	B0 000	720 896
0B 000	45 056	C0 000	786 432
0C 000	49 152	D0 000	851 968
0D 000	53 248	E0 000	917 504
0E 000	57 344	F0 000	983 040
0F 000	61 440	100 000	1 048 576
10 000	65 536	200 000	2 097 152
11 000	69 632	300 000	3 145 728
12 000	73 728	400 000	4 194 304
13 000	77 824	500 000	5 242 880
14 000	81 920	600 000	6 291 456
15 000	86 016	700 000	7 340 032
16 000	90 112	800 000	8 388 608
17 000	94 208	900 000	9 437 184
18 000	98 304	A00 000	10 485 760
19 000	102 400	B00 000	11 534 336
1A 000	106 496	C00 000	12 582 912
1B 000	110 592	D00 000	13 631 488
1C 000	114 688	E00 000	14 680 064
1D 000	118 784	F00 000	15 728 640
1E 000	122 880	1 000 000	16 777 216
1F 000	126 976	2 000 000	33 554 432

Una parte frazionaria esadecimale può essere convertita in decimale, nel modo seguente:

1. Trasformate la parte frazionaria in intero moltiplicandola per 16^n, dove n è il numero di cifre significative a destra del punto. Esempio:

$$0.\,CA9BF3_{16} = CA9\,BF3_{16} \times 16^{-6}$$

2. Trasformate questo valore in decimale intero:

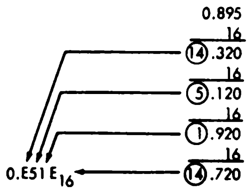
$$CA9\,BF3_{16} = 13\,278\,195_{10}$$

3. E quindi moltiplicatelo per 16^{-6} ($16^{-6} = 596046448 \times 10^{-16}$):

$$\begin{array}{r} 13\,278\,195 \\ \times\ 596\,046\,448 \times 10^{-16} \\ \hline 0.791\,442\,096_{10} \end{array}$$

Le parti frazionarie decimali possono essere convertite in esadecimale mediante moltiplicazioni successive, della parte decimale, per 16_{10}.
Dopo ogni moltiplicazione la parte intera che ne risulta viene rimossa per costituire la parte frazionaria esadecimale. Prima è necessario però convertire questo valore parziale in esadecimale.

Esempio: convertire 0.895_{10} in esadecimale.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
01	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
02	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
03	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
04	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
05	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
06	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
07	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
08	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
09	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255

CONVERSIONE DI INTERI TRA DECIMALE ED ESADECIMALE (Continua)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
11	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
12	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
13	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
14	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
15	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
16	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
17	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
18	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
19	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511
20	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
21	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
22	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
23	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
24	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
25	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
26	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
27	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
28	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
29	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
30	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
31	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
32	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
33	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
34	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
35	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
36	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
37	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
38	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
39	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

CONVERSIONE DI INTERI TRA DECIMALE ED ESADECIMALE (Continua)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
41	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
42	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
43	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
44	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
45	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
46	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
47	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
48	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
49	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
50	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
51	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
52	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
53	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
54	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
55	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
56	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
57	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
58	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
59	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
60	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
61	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
62	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
63	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
64	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
65	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
66	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
67	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1562	1663
68	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
69	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791

CONVERSIONE DI INTERI TRA DECIMALE ED ESADECIMALE (Continua)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
70	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
71	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
72	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
73	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
74	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
75	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
76	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
77	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
78	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
79	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047
80	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
81	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
82	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
83	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
84	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
85	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
86	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
87	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
88	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
89	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
90	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
91	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
92	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
93	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
94	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
95	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
96	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
97	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
98	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
99	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

CONVERSIONE DI INTERI TRA DECIMALE ED ESADECIMALE (Continua)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A0	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A1	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A2	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A3	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A4	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A5	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A6	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A7	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A8	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A9	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B0	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B1	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B2	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B3	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B4	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B5	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B6	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B7	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B8	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B9	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071
C0	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C1	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C2	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C3	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C4	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C5	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C6	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C7	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C8	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C9	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327

CONVERSIONE DI INTERI TRA DECIMALE ED ESADECIMALE (Continua)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
D0	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D1	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D2	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D3	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D4	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D5	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D6	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D7	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D8	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D9	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583
E0	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E1	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E2	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E3	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E4	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E5	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E6	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E7	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E8	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E9	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F0	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F1	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F2	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F3	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F4	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F5	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F6	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F7	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F8	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F9	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

APPENDICE K

BIBLIOGRAFIA

BASIC

Albrecht, Finkle, and Brown. *BASIC.* Peoples Computer Company, Menlo Park, California, 1967.

Coan, James S. *Advanced BASIC.* Hayden Book Co., Rochelle Park, New Jersey.

Coan, James S. *Basic BASIC.* Hayden Book Company, Rochelle Park, New Jersey.

Dwyer, T. *A Guided Tour of Computer Programming in BASIC.* Houghton Mifflin Company, 1973.

Kemeny, J., and Kurtz, T. *BASIC Programming.* Peoples Computer Company, Menlo Park, California, 1967.

Pegels, C. *BASIC: A Computer Programming Language.* Holden-Day, Inc., 1973.

Peoples Computer Company. *What to Do After You Hit Return.* Menlo Park, California.

Sack, J., and Meadows, J. *Entering BASIC.* Science Research Associates, 1973.

Programmazione in linguaggio Assembly

Leventhal, Lance A. *6502 Assembly Language Programming.* Osborne/McGraw-Hill, Berkeley, California, 1979.

Osborne, A. *An Introduction to Microcomputers: Volume 1 — Basic Concepts.* 2nd ed., Osborne/McGraw-Hill, Berkeley, California, 1980.

Scanlon, Leo J. 6502 *Software Design.* Howard W. Sams, Indianapolis, Indiana.

Zaks, Rodnay. *6502 Applications Book.* Sybex, Berkeley, California.

Periodici

"Apple-Cart," *Creative Computing*. P.O. Bo z 789-M, Morristown, New Jersey 07960.

Micro, P.O Box 6502, Chelmsford, Massachusetts 01824.

Nibble. P.O. Box 325, Lincoln, Massachusetts 01773.

Purser's Magazine. P.O. Box 466, El Dorado, California 95623.

Softalk. 10432 Burbank Bl., N. Hollywood, California 91601.

Pubblicazioni Apple

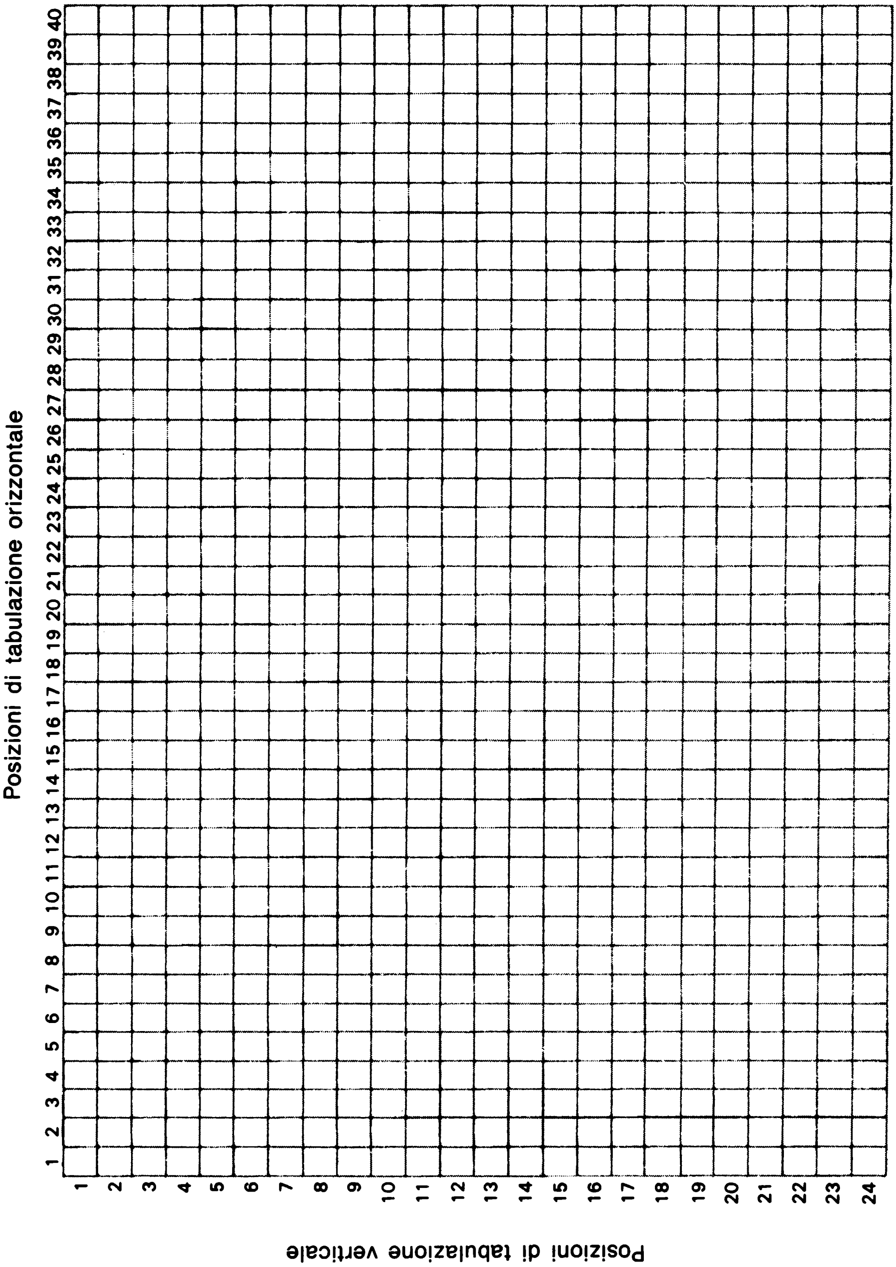
Apple II/II Plus Reference Manual	A2L0001A
Parallel Printer Interface Manual	A2L0004
Apple II BASIC Programming Manual	A2L0005
Applesoft II Refernce Manual	A2L0006
Communications Interface Manual	A2L0007
High-Speed Serial Interface Manual	A2L0008
Disk II Floppy Disk Manual (DOS 3.2.1)	A2L0012
Applesoft Tutorial Manual	A2L0018
Graphics Tablet Manual	A2L0033
Silentype Manual	A2L0034
The DOS Manual (DOS 3.3)	A2L0036

APPENDICE L

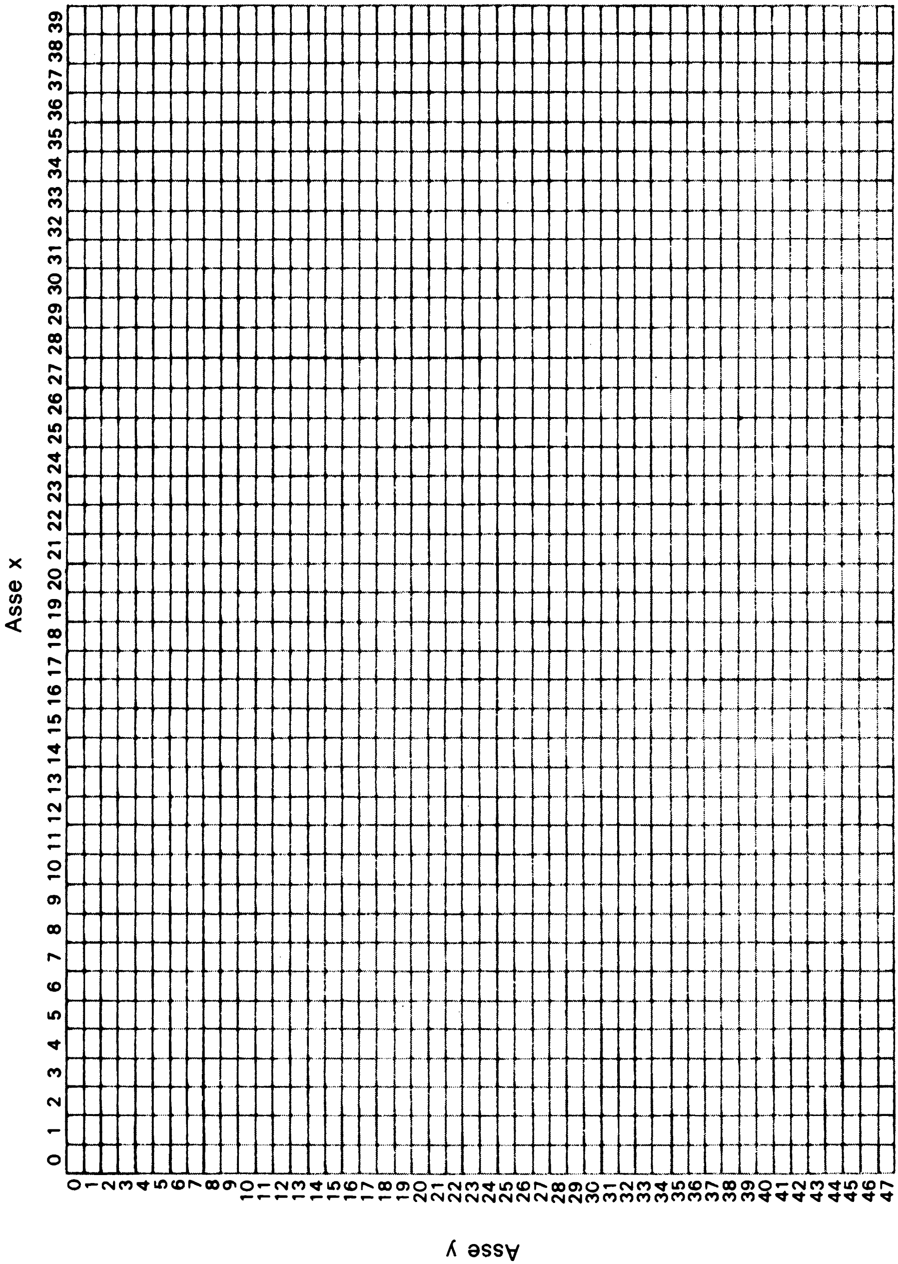
SCHEMATIZZAZIONE DELLO SCHERMO

Gli schemi che riportiamo qui di seguito vi possono essere utili per preparare delle corrette visualizzazioni sullo schermo.

Nel caso del modo testo la prima riga e la prima colonna hanno il numero 1. Nel caso invece del modo grafico in bassa risoluzione la prima riga e la prima colonna hanno il numero 0.



Schermo nel modo testo



Schermo grafico a bassa risoluzione

L'AVT è un computer Apple-compatibile. Le uniche differenze di tastiera tra i due sistemi vengono riportate nella semplice tabella di conversione seguente.

CROSS REFERENCE OF KEY-BOARD FUNCTION

DESCRIPTION	APPLE PLUS SYSTEM	AVT-II SYSTEM	REMARK
CLEAR SCREEN	ESC-@ (SHIFT)	HOME, ESC-@ ESC-P (SHIFT)	AVT USES 1 CHARACTER COMMAND
ESCAPE CURSOR UP	ESC-I	UP ARROW KEY, CTRL-K ESC-I (SHIFT)	AVT USES 1 CHARACTER COMMAND
ESCAPE CURSOR DOWN	ESC-M	DOWN ARROW KEY ESC-I (SHIFT)	AVT USES 1 CHARACTER COMMAND
[ˆ]	N (SHIFT)	ˆ, N (SHIFT)	
[[ ]	NONE	[	
[ ]]	M (SHIFT)	]	
BACK SPACE	CTRL-H LEFT ARROW KEY	CTRL-H, BS, DEL LEFT ARROW KEY	
˜, {,}, `, :	NONE	NONE	TO BE ANNOUNCED
RESET	RESET	BRAKE	
BOTTOM LINE_	CTRL-O	<u>D</u>EL	

Se possedete un Apple e volete conoscerlo a fondo, se volete comprarlo, o se semplicemente volete imparare la sua programmazione, non dovrete più affannarvi nella ricerca su "mille" testi di tutte le informazioni necessarie.
Risparmiando tempo e fatica troverete in questo libro, vero strumento di programmazione, tutte le risposte, comprese alcune che vi occorrono per una perfetta operatività del sistema, vere "primizie".
Conoscerete i vari componenti del sistema e come usarli al meglio. Sarete già a questo punto in grado di utilizzare molti dei programmi (tutti provati) scritti per qualunque applicazione: commerciale, finanziaria e hobbistica.
Verrete guidati alla programmazione in due versioni BASIC (Apple-soft e integer BASIC), e a usare per creare emozionanti effetti le caratteristiche grafiche e sonore del sistema. (Alcune nuove subroutine vi permetteranno di usare l'alta risoluzione grafica).
Imparerete a memorizzare su disco sia programmi che archivi dati, come ad inserire un programma scritto in assembler in uno scritto in BASIC.
Vedrete minuziosamente descritto il programma supervisore Monitor nelle due versioni: standard e Autostart.
E poi ancora, tutte le istruzioni e funzioni BASIC, e ben 12 Appendici veramente basilari: indirizzi utili di PEEK e POKE, parole riservate, impiego della memoria...

L. 26.000 Cod. 331P ISBN 88/7056/128-3

GRUPPO
EDITORIALE
JACKSON

Lon Poole
Martin McNiff
e
Steven Cook

APPLE III® GUIDA ALL'USO

73